Algebraic Theories for Name-Passing Calculi

Joachim Parrow[*] Davide Sangiorgi[†]

May 11, 1994

Abstract

In a theory of processes the *names* are atomic data items which can be exchanged and tested for identity. A well-known example of a calculus for name-passing is the π-calculus, where names additionally are used as communication ports. We provide complete axiomatisations of late and early bisimulation equivalences in such calculi. Since neither of the equivalences is a congruence we also axiomatise the corresponding largest congruences. We consider a few variations of the signature of the language; among these, a calculus of deterministic processes which is reminiscent of sequential functional programs with a conditional construct. Most of our axioms are shown to be independent. The axiom systems differ only by a few simple axioms and reveal the similarities and the symmetries of the calculi and the equivalences.

Work supported by the ESPRIT BRA project 6454 "CONFER".

---

[*] Swedish Institute of Computer Science, Uppsala University, and The Royal Institute of Technology, Sweden

[†] LFCS, Department of Computer Science, University of Edinburgh, U.K.

1 Introduction

Consider the following two programs:

```
P1:  input x;                          P2:  input x;
     if x = 3 then output x;                if x = 3 then output 3;
```

It is obvious that these programs have the same behaviour in the sense that their possible interactions with an environment (through `input` and `output` statements) are the same. The purpose of this paper is to provide algebraic laws, i.e., syntactic transformation rules, whereby such equalities can be established. This requires a careful definition of what it means for two things to have the "same behaviour", and since we aspire to a complete proof system it also means that we must concentrate on a limited but interesting set of programming constructs. In both these respects we will investigate a couple of different varieties; thus we will obtain several interrelated sets of algebraic laws. A main motivation is to provide a cleaner understanding of the constructs through the algebraic properties they satisfy. Moreover, equational theories (and the proof of their completeness) facilitate reasoning about programs and can be the formal basis for computer aided transformations in practical applications.

We will use a language with primitives for input, output, nondeterminism, parallelism, restriction and conditional choice. Values may be transmitted in interactions with the environment and between components; however, the programs may not perform any computation directly on the values apart from testing equality. The inspiration for our formalisation comes from the π-calculus [MPW92], where it has been demonstrated that such a language can encode computations over arbitrary data domains. For example `P1` and `P2` above would be written

$$\texttt{P1}: \quad i(x).\,[x=3]\overline{o}x \qquad\qquad \texttt{P2}: \quad i(x).\,[x=3]\overline{o}3$$

Here $i(x)$ means "`input x`", $[x=3]$ means "`if x=3 then`", and $\overline{o}x$ and $\overline{o}3$ mean "`output x`" and "`output 3`", respectively. The operational semantics of `P1` says that it has a transition $\xrightarrow{i(x)}$, binding x, and resulting in $[x=3]\overline{o}x$. The agent `P2` has a similar transition, leading to $[x=3]\overline{o}3$. Here i, o, x and 3 are *names* and `P1` and `P2` are *agents*. Although the names are used for different purposes (i and o designate ports, x a value variable and 3 a value constant) the π-calculus makes no formal distinction between these kinds, so port names can be transmitted in interactions between agents. Together with constructs for parallelism and scope restriction this lends the calculus a considerable expressive power ([Mil92, San92]). In the present paper we will not be much concerned with expressiveness, and the fact that port names can be considered as data objects turns out to be non-essential for our purposes. Our algebraic characterisations are thus also relevant for more limited versions of value-passing calculi such as CCS [Mil89], CSP [Hoa85] and ACP [BK85]; moreover the axiomatisation of the subcalculus without nondeterminism is in some respects similar to axiomatisations of functional programs. An extended discussion of related work on axiomatisations of process calculi and on functional programs is deferred to the conclusion. In the rest of this introduction we shall explain the novelty of our approach.

A standard way to introduce computations on a domain of values is to use a set of value variables and let programs, or agents, with free variables be interpreted as functions from "environments" (mappings from variables to values) to behaviours. The operational semantics and associated behavioural equivalence is defined for ground agents (without free variables) and is extended to arbitrary agents as equivalence in all environments. As remarked by Milner et al [MPW92] this distinction between "variable" and "constant" is sometimes artificial — a name can in some contexts be considered a variable and in other contexts a constant. We will not repeat these arguments here, but we adopt the convention from the π-calculus to use only one syntactic category of names which can function as both variables and constants. As a consequence there will be no distinguished ground agents, so the operational semantics must cater for agents corresponding to programs with free variables. As an example, take a subagent of `P1`:

$$[x=3]\overline{o}x$$

(which can be paraphrased `if x=3 then output x`). Which transitions has this agent got? Here x and 3 are just names, but they are not the *same* name. Therefore the condition "$x = 3$" is false, for the same reason that e.g. "$4 = 3$" is false, and the agent has no transitions at all — it is in fact equivalent (in a sense to be made precise later) to $\mathbf{0}$, an agent lacking transitions. Thus, writing $\dot{\sim}$ for equivalence, we have

$$[x=3]\overline{o}x \;\dot{\sim}\; \mathbf{0} \tag{1}$$

But from this we can*not* conclude that $i(x).\,[x=3]\overline{o}x \;\dot{\sim}\; i(x).\,\mathbf{0}$. These agents begin by doing an input transition $\xrightarrow{i(x)}$, and equivalence will require that the transitions from one should be mimicked by the other for all possible instantiations of the bound name x. Here an instantiation is just a substitution of names for names, so we would require that (1) holds for arbitrary substitutions of x. Consider the substitution which sends x to 3; this leaves $\mathbf{0}$ unaffected (since $\mathbf{0}$ contains no names) but it transforms the left hand side to $[3=3]\overline{o}3$. Since the condition "$3 = 3$" holds this agent has a transition $\xrightarrow{\overline{o}3}$ and cannot be equivalent to $\mathbf{0}$.

This small example shows that equivalence is not in general substitutive under the scope of an input construct. In spite of this it can be axiomatised; an inference rule for input prefixing will be, approximately,

$$\text{If } P\{y/x\} \mathrel{\dot{\sim}} Q\{y/x\} \text{ for all } y, \text{ then } i(x).\,P \mathrel{\dot{\sim}} i(x).\,Q \tag{2}$$

This rule essentially invokes a case analysis for names bound by input. For example, consider the following two subterms of `P1` and `P2`:

$$[x=3]\overline{o}x \;\;\dot{\sim}\;\; [x=3]\overline{o}3 \tag{3}$$

Clearly (3) holds since both sides are equivalent to **0**, but we can also prove that (3) holds when any name is substituted for x as follows: if that name is 3 then both sides reduce to $\overline{o}3$, if not both sides reduce to **0**. Therefore (2) can be applied to conclude `P1` $\dot{\sim}$ `P2`. However, in this way more substantial proofs tend to be awkward, containing massive case analyses. Note that the same case analyses may occur in a calculus where value variables are distinguished, at least if the proof system is defined for ground terms only (cf. the system by Hennessy and Ingólfsdóttir [HI89]). For to prove (3) in a calculus where x is a value variable, that equation must be established for all environments, i.e. all possible substitutions of values for x.

In order to avoid the case analyses we must eliminate the culprit that generates them, namely (2). One way to achieve this is to conduct the proofs for some congruence included in the equivalence. A good candidate is the largest such congruence, written $\sim$, which is simply equivalence under all possible substitutions. This characterisation lends no improvement to the proof system (since it just refers back to equivalence), so an alternative proof system for congruence is needed. A main contribution of the present paper is to establish such a system. To see an elementary application of it, one of our laws is

$$[x=y]\,\alpha.\,P \;\;\sim\;\; [x=y]\,(\alpha\{x/y\}).\,P$$

for arbitrary names x, y (here $\{x/y\}$ means substitution of x for y). Using this law we can obtain

$$[x=3]\overline{o}x \;\;\sim\;\; [x=3](\overline{o}x\{3/x\}) \;\;=\;\; [x=3]\overline{o}3$$

and from this we immediately get `P1` $\sim$ `P2` (without any case analysis) since $\sim$ is a congruence.

We will adopt the notion of (strong) bisimulation as the notion of "same behaviour"; this is one of the most studied equivalence concepts in process calculi, and for deterministic agents it coincides with trace equivalence. For two agents P and Q to be equivalent, each transition from P must be mimicked by Q (and vice versa), leading again to equivalent agents. Previous work on the π-calculus has shown that the strong bisimulation equivalence is less straightforward when value-passing is incorporated as a basic notion. Briefly stated, strong bisimulation proliferates into four distinct equivalences: there is a *late* and an *early* variety, and for the reason mentioned above neither of these is preserved by input prefix so it is also of interest to consider the corresponding congruences.

The proliferation to late and early is a consequence of the interaction between value-passing and nondeterminism, and seems to be specific to bisimilarity — it is not present in e.g. testing or trace equivalences [BD92, He91]. The early bisimulation equivalence is the one obtained by Milner [Mil89] by translating full CCS, i.e., CCS including value-passing, into infinitary pure CCS, i.e., CCS without value-passing but with infinite summation. In this translation the transmission of an object on a port is regarded as one atomic event; correspondingly the early bisimulation requires that bisimilar agents can match each other for such events. The early bisimulation is also the one which naturally arises from a reduction semantics on the lines of Berry and Boudol's Chemical Abstract Machine [BB90], as shown in [San92] using the notion of barbed bisimulation.

The late bisimulation equivalence builds on a more refined operational intuition: an agent can decide to receive input on a port, and by doing so it becomes a function from values to agents. There are thus two atomic "events" corresponding to an input transition, namely first committing on a port and then instantiating the function with the received value. Late bisimulation puts the stronger requirement on equivalent agents that these atomic events are precisely matched. Thus late equivalence is strictly finer than early (but for deterministic agents both early and late coincide with trace equivalence). Further discussion of the early vs. late question is in [MPW91], where the equivalences are given modal logic characterisations. Late bisimulation equivalence was axiomatised in the original paper on the π-calculus [MPW92] but no axiomatisation of early equivalence has been published. In the present paper we will give one axiom which, when added to the late axiomatisation, gives a complete system for early.

Since our language draws its primitives from the π-calculus it can be thought of as an extension of CCS. We will only study the finitary part, omitting the constructs for recursion and replication (with these constructs all equivalences are non r.e. and thus cannot have decidable axiomatisations). In one important respect we go beyond the π-calculus: we include a binary conditional construct "`if` φ `then` P `else` Q", where φ is a boolean expression built from the standard boolean connectives and a *matching* construct for equality of names. This contrasts with the original formulation of the π-calculus, which only has the unary matching construct "`if` $x = y$ `then` P". Our choice is motivated by two reasons. Firstly, we need to express *mismatching*, or inequalities of names, of the form "`if` $x \neq y$ `then` P"; we have not been able to axiomatise the two early equivalences or the late congruence without it. Secondly, we hope to gain in clarity and generality. We would like to present our results as not strictly specific to π-calculus; the conditional is a familiar construct in programming languages and it is interesting to see how it interacts with the other operators.

The resemblance with functional programming languages is made clearer by omitting the operator for nondeterministic choice (the CCS summation, "+"). While this (or some similar operator) appears indispensable for axiomatising the parallel composition in an interleaving semantics, it is an alien device in functional programming practice (parallel programming languages like Ada and Occam do have choice constructs, but it is debatable whether they are semantically similar to "+"). We will therefore also explore a deterministic subcalculus which has several points of interest. The subcalculus must deal extensively with binary conditionals (`if` ... `then` ... `else` ...) where previously we could reduce them to sums of unary conditionals (`if` ... `then` ...). The completeness proof must consequently take a new direction. The resulting axiom system is reminiscent of basic theories of conditionals such as McCarthy's [McC63], and a comparison with the system for the full calculus reveals that the laws for conditional and summation are structurally similar. The use of binary conditionals also indicates that our choice of axioms may be relevant for other dialects of process algebras with different summation operators. For example, in CSP [Hoa85] summation is replaced by internal and external nondeterminism, and with these it is less obvious how to reduce binary conditionals to unary.

The rest of the paper is structured as follows. Section 2 contains preliminary definitions of agents and equivalences. We concentrate here on a language without parallelism and scope restriction. This smaller language already shows all major obstacles for the axiomatisations and hence makes the presentation of our ideas neater. Although our definitions will be formally self-contained, a reader may find an introductory exposition to the π-calculus [MPW92] useful to gain intuition. In Section 3 we examine axiomatisations of bisimilarities; we first recapitulate the axiomatisation of late bisimilarity and then show how a single axiom extends this to early. Section 4 contains the axiomatisations of the induced congruences. Basically, the axioms for a congruence (early or late) are obtained from those for the corresponding bisimulation by adding a few laws for manipulating conditionals. Remarkably, the difference between late and early congruence is again expressed by the same axiom.

We then turn to consider variants of the language. In Section 5 we restrict attention to deterministic agents, obtained by omitting the construct for nondeterminism. Here the late and early equivalences coincide but the congruence is still different from the equivalence. Essentially, the axiomatisations are obtained by omitting all axioms mentioning nondeterminism. In Section 6 we show that in each axiom system the axioms are necessary, i.e. they are independent of the other axioms. The comparatively straightforward independence proofs increase our confidence in the choice of axiom systems. In Section 7 we show how the language is extended with primitives for parallelism and scope restriction. The effect on the axiomatisations is to add enough laws to eliminate, or push as deep as possible, all occurrences of these operators. The same set of laws can be used with any equivalence. Finally Section 8 contains directions for further research and more comparisons with related work.

2 Agents and Equivalences

2.1 Syntax

Assume a set of *port names* ranged over by $a, b, \ldots$, a set of *object names* ranged over by u, v, x, y, z. These sets are not necessarily distinct — for example in the π-calculus they coincide as the set of names. The *prefixes*, ranged over by α, β, are given by:

$$\begin{array}{rcll} \alpha & ::= & a(x) & \text{(input)} \\ & \mid & \overline{a}x & \text{(output)} \end{array}$$

In these a is called the *port* and x the *object* of the prefix. The set of *free names* $\mathrm{fn}(\alpha)$ is $\{a, x\}$ in the output prefix and $\{a\}$ in the input prefix; the set of *bound names* $\mathrm{bn}(\alpha)$ is $\emptyset$ in the output prefix and $\{x\}$ in the input prefix.

The *conditions*, ranged over by φ, ψ etc, are given by the following grammar:

$$\begin{array}{rcll} \varphi & ::= & x = y & \text{(name matching)} \\ & \mid & \neg\varphi & \text{(negation)} \\ & \mid & \varphi \vee \psi & \text{(disjunction)} \end{array}$$

We will use *True* to abbreviate the condition $x = x$ for some x, and *False* to abbreviate $\neg\,$*True*; further we write $x \neq y$, called *mismatching*, to abbreviate $\neg(x = y)$. As usual we let $\wedge$ be defined by $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$.

Definition 2.1 *The set of* agents *is defined as follows (we use P, Q, R range over agents):*

$$
\begin{array}{rcll}
P & ::= & \mathbf{0} & \textit{(inaction)} \\
 & | & \alpha\,.\,P & \textit{(prefix)} \\
 & | & \varphi\,P\,Q & \textit{(conditional)} \\
 & | & P+Q & \textit{(summation)}
\end{array}
$$

The output prefix construct $\overline{a}x\,.\,P$ will, intuitively, mean "first transmit the object x on the port a and then do P." The input prefix $a(x)\,.\,P$ will mean "input something for x on the port a and then do P." Summation $P+Q$ will mean "behave as one of P and Q", and the conditional $\varphi\ P\ Q$ will mean "if φ then P else Q". The order of precedence among the operators is the order listed above. We will frequently use parentheses around agents and square brackets around conditions to facilitate reading. We use the *unary conditional* $\varphi\,P$ to abbreviate $\varphi\ P\ \mathbf{0}$. A conditional consisting of only one matching or one mismatching is called an *elementary conditional.* (Through the laws presented later in this paper, any term can be rewritten to a term where all conditionals are unary and elementary.) We further use the abbreviation $\sum_{i=1}^{n} P_i$ to mean $P_1+\cdots+P_n$ if $n>0$ or $\mathbf{0}$ if $n=0$. Sometimes we will omit a trailing "$.\,\mathbf{0}$", so e.g. $\alpha+\beta$ will mean $\alpha\,.\,\mathbf{0}+\beta\,.\,\mathbf{0}$.

Each occurrence of x in $a(x)\,.\,P$ is a *bound* occurrence, and an occurrence of a name in an agent is *free* if it is not bound. The set of free object names in P is written $\mathrm{fn}(P)$, and we sometimes write $\mathrm{fn}(P,Q,\ldots,x,y,\ldots)$ as an abbreviation for $\mathrm{fn}(P)\cup\mathrm{fn}(Q)\cup\ldots\cup\{x,y,\ldots\}$. Similarly $\mathrm{n}(P)$ and $\mathrm{n}(\varphi)$ stand for all names (free or bound) in P and φ. Note that $\mathrm{fn}(P)$ and $\mathrm{n}(P)$ by definition are finite sets.

A *substitution* is a function from names to names. We will use the normal notation for substitutions; e.g., $\{x/y\}$ is the function which sends y to x and is identity on all names but y. We use σ,ρ etc. to range over substitutions, and write $P\sigma$ for the agent obtained from P by replacing all free occurrences of any name x by $\sigma(x)$, with change of bound object names if necessary to avoid captures. Similarly $\alpha\sigma$ (or $\varphi\sigma$) is the result of applying σ to the action α (or condition φ), and does not affect a bound name in α if any. Substitutions have precedence over the operators of the language; $\sigma\rho$ is the composition of substitution where σ is performed first; therefore $P\sigma\rho$ is $(P\sigma)\rho$.

2.2 Transitional Semantics

We use $[\![\varphi]\!]$ to denote the evaluation of φ into the ordinary two-valued boolean domain $\{\mathit{True}, \mathit{False}\}$, inductively defined in the standard way:

$$\begin{array}{rcll} [\![x = x]\!] & = & \mathit{True} & \\ [\![x = y]\!] & = & \mathit{False} & \quad \text{if } x \neq y \\ [\![\neg\varphi]\!] & = & \neg[\![\varphi]\!] & \\ [\![\varphi \vee \psi]\!] & = & [\![\varphi]\!] \vee [\![\psi]\!] & \end{array}$$

A *transition* is of the form

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

where α is a prefix, i.e. $\overline{a}x$ or $a(x)$, which we call the *action* of the transition. Intuitively, $P \xrightarrow{\alpha} Q$ means that P can evolve into Q, and in doing so perform α.

Definition 2.2 *The transitions between agents are the transitions which can be inferred from the following rules.*

$$\frac{-}{\alpha . P \xrightarrow{\alpha} P}$$

$$\frac{[\![\varphi]\!] = \mathit{True}, \quad P \xrightarrow{\alpha} P'}{\varphi\, P\, Q \xrightarrow{\alpha} P'} \qquad \frac{[\![\varphi]\!] = \mathit{False}, \quad Q \xrightarrow{\alpha} Q'}{\varphi\, P\, Q \xrightarrow{\alpha} Q'}$$

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

For the purposes of the transitional semantics we will not distinguish between alpha-equivalent agents, i.e., agents which only differ in the choice of bound names. Formally, we can either define the transitions on alpha-equivalence classes of agents, or add the rule

$$\frac{P' \xrightarrow{\alpha} Q \qquad P,\, P' \text{ alpha-convertible}}{P \xrightarrow{\alpha} Q}$$

Issues related to alpha-equivalence are treated in depth in previous work on the π-calculus; we will ignore them in the present paper.

2.3 Bisimilarities and Congruences

Late and early bisimulations are defined as follows.

Definition 2.3 *A binary relation $\mathcal{S}$ on agents is a* late simulation *if $P\mathcal{S}Q$ implies that*

1. *If $P \xrightarrow{\overline{a}x} P'$, then for some Q', $Q \xrightarrow{\overline{a}x} Q'$ and $P'\mathcal{S}Q'$.*
2. *If $P \xrightarrow{a(x)} P'$ and $x \notin \mathrm{fn}(P, Q)$, then for some Q', $Q \xrightarrow{a(x)} Q'$ and for all y, $P'\{y/x\}\mathcal{S}Q'\{y/x\}$.*

The relation $\mathcal{S}$ is a late bisimulation *if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are late simulations. Two agents P and Q are* late bisimilar, *written $P \overset{\cdot}{\sim}_{\mathrm{L}} Q$, if $P\mathcal{S}Q$ for some late bisimulation $\mathcal{S}$.* $\square$

Thus late bisimilarity requires of input transitions that a matching transition exists which is adequate for all instantiations of the object. If the quantifiers in clause 2 are commuted we obtain early bisimilarity, which has the weaker requirement that for any instantiation there is an adequate transition:

Definition 2.4 *A binary relation $\mathcal{S}$ on agents is an* early simulation *if $P\mathcal{S}Q$ implies that*

1. *If $P \xrightarrow{\overline{a}x} P'$, then for some Q', $Q \xrightarrow{\overline{a}x} Q'$ and $P'\mathcal{S}Q'$.*
2. *If $P \xrightarrow{a(x)} P'$ and $x \notin \mathrm{fn}(P, Q)$, then for all y there exists Q' s.t. $Q \xrightarrow{a(x)} Q'$ and $P'\{y/x\}\mathcal{S}Q'\{y/x\}$.*

The relation $\mathcal{S}$ is an early bisimulation *if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are early simulations. Two agents P and Q are* early bisimilar, *written $P \overset{\cdot}{\sim}_{\mathrm{E}} Q$, if $P\mathcal{S}Q$ for some early bisimulation $\mathcal{S}$.* $\square$

We omit the straightforward proofs that both late and early bisimilarity are equivalences, and that they are preserved by all operators except input prefix. Late bisimilarity is extensively studied in [MPW92]. Every late bisimulation is an early bisimulation, hence $\overset{\cdot}{\sim}_{\mathrm{L}} \subseteq \overset{\cdot}{\sim}_{\mathrm{E}}$. The following example shows that the inclusion is strict: Let R be not equivalent with $\mathbf{0}$, and

$$P = a(x).\, R + a(x).\, \mathbf{0} \qquad \text{and} \qquad Q = P + a(x).\, [x = y]\, R \tag{4}$$

Then $P \overset{\cdot}{\sim}_{\mathrm{E}} Q$, but $P \overset{\cdot}{\not\sim}_{\mathrm{L}} Q$.

As remarked in the introduction neither of these equivalences are preserved by substitution of names. For instance, using $\overset{\cdot}{\sim}$ to range over $\overset{\cdot}{\sim}_{\mathrm{L}}$ and $\overset{\cdot}{\sim}_{\mathrm{E}}$, if $x \neq y$ we have

$$[x = y]\, \alpha.\, \mathbf{0} \overset{\cdot}{\sim} \mathbf{0}$$

since neither of the agents has any transition. But this is false after substituting y for x; hence

$$a(x).\, [x = y]\, \alpha.\, \mathbf{0} \;\overset{\cdot}{\not\sim}\; a(x).\, \mathbf{0}$$

To obtain congruences we have to require bisimilarity over all substitutions.

Definition 2.5 *Two agents P and Q are* late congruent *(resp.* early congruent*), written $P \sim_{\mathrm{L}} Q$ (resp. $P \sim_{\mathrm{E}} Q$), if for all substitutions σ, it holds that $P\sigma \overset{\cdot}{\sim}_{\mathrm{L}} Q\sigma$ (resp. $P\sigma \overset{\cdot}{\sim}_{\mathrm{E}} Q\sigma$).*

For the proof that $\sim_{\mathrm{L}}$ and $\sim_{\mathrm{E}}$ are congruence relations, see [MPW92] or [MPW91]. The strict inclusion $\dot{\sim}_{\mathrm{L}} \subset \dot{\sim}_{\mathrm{E}}$ is maintained on the congruences and we have $\sim_{\mathrm{L}} \subset \sim_{\mathrm{E}}$ (the agents P and Q in (4) showing the strictness of the former inclusion can also be used to demonstrate the strictness of the latter).

3 Late and Early Bisimilarity

3.1 Nomenclature

In the rest of this paper we will study axiom systems for the four equivalences and for a variety of calculi. Many of the laws will recur in several systems. To facilitate presentation we will give each law a name and use that name consistently throughout the paper. The name of a law will be one or two letters, possibly followed by a number, and possibly followed by an asterisk. The letters refer to the main operators mentioned in a law; for example, "**C**" designates a law for conditionals, "**SC**" a law for sums of conditionals, and "**CC**" a law for nested conditionals. Whenever there is more then one law named by the same letter(s) we use numbers to distinguish between them (numbers are assigned in order of appearance in this paper). An asterisk is used to signify a stronger variant of the law with binary rather than unary conditionals. Since we will axiomatise some equivalences which are not congruences the substitutive properties of an equivalence will always be stated explicitly by the axiom system. The corresponding inference rules are labelled **I** followed by a letter signifying the main operator. In all laws we use P, Q, R as meta variables to represent arbitrary agents, a, x, y to represent arbitrary names and φ, ψ to represent arbitrary conditions. For ease of reference the Appendix at the end of this paper contains a table of all laws (note that some laws are not valid in all systems).

The following law for alpha-conversion will be tacitly present in all axiom systems:

A If P and Q are alpha-equivalent, then $P = Q$

In view of this we will not distinguish between alpha-equivalent agents, and we adopt $\equiv$ for syntactical identity between agents.

If $\mathcal{AS}$ is an axiom system we write $\mathcal{AS} \vdash P = Q$, or sometimes $P \overset{\mathcal{AS}}{=} Q$, if $P = Q$ can be inferred from $\mathcal{AS}$ using equational reasoning, that is, the fact that $=$ is an equivalence. When demonstrating inferences we shall write $P \overset{\mathbf{Zi,Zj}}{=} Q$ to mean that Q is derived from P using the axioms **Zi** and **Zj** (possibly using each of them more than once), plus the inference rules.

IP1	If $P = Q$ then $\overline{a}x.\,P = \overline{a}x.\,Q$	
IP2	If $P\{y/x\} = Q\{y/x\}$ for all $y \in \mathrm{fn}(P, Q, x)$ then $a(x).\,P = a(x).\,Q$	
IS	If $P = Q$ then $P + R = Q + R$	
S1	$P + \mathbf{0} = P$	
S2	$P + P = P$	
S3	$P + Q = Q + P$	
S4	$P + (Q + R) = (P + Q) + R$	
C1	$\varphi\, P\, Q \;=\; P$	if $[\![\varphi]\!] = \mathit{True}$
C2	$\varphi\, P\, Q \;=\; Q$	if $[\![\varphi]\!] = \mathit{False}$

Table 1: Axiom system $\mathcal{LB}$ for late bisimilarity.

3.2 Late Bisimilarity

The axiom system $\mathcal{LB}$ for late bisimilarity is given in Table 1. The universal quantification in **IP2** is necessary to be able to handle input prefixes since this operator does not preserve bisimilarity. On the other hand, conditionals preserve bisimilarity but a separate inference law for this is unnecessary in view of **C1** – **C2**. Observe that a non-injective substitution may affect the applicability of **C1** and **C2**, so although these axioms can reduce any top-level conditional they do not in general admit removals of conditionals under the scope of an input prefix.

Proposition 3.1 (soundness of $\mathcal{LB}$) *If* $\mathcal{LB} \vdash P = Q$ *then* $P \stackrel{.}{\sim}_{\mathrm{L}} Q$.

PROOF: By exhibiting the appropriate bisimulations — see [MPW92]. □

The converse is established through the concept of head normal forms and an induction over the *depth* $d(P)$ of an agent P, which is the maximal number of nested prefix operators, i.e.,

$$\begin{aligned} d(\mathbf{0}) &= 0, \quad d(\alpha.\,P) \;=\; 1 + d(P), \\ d(\,\varphi\, P\, Q\,) &= d(P + Q) \;=\; \max(d(P), d(Q)) \end{aligned}$$

An agent P is in *head normal form* (hnf in short) if it is a sum of prefixes, i.e.

$$P \equiv \sum_{i=1}^{n} \alpha_i.\,P_i$$

Lemma 3.2 *For any agent P there is a hnf Q of no greater depth s.t. $\mathcal{LB} \vdash P = Q$.*

Proof: Trivial by induction on the structure of P. Inaction and prefix forms are hnf's, and sums and conditionals can be rewritten by **S1**, **S4**, **C1**, and **C2**. □

Theorem 3.3 (completeness of $\mathcal{LB}$ for $\dot\sim_{\mathrm{L}}$) *If $P \dot\sim_{\mathrm{L}} Q$ then $\mathcal{LB} \vdash P = Q$.*

Proof: A variant of the proof is contained in [MPW92] but it will be instructive to repeat the outlines here. The proof is by induction on the depths of P and Q. By the preceding lemma we can assume that P and Q are in hnf. The base case of the induction is trivial since **0** is the only hnf of depth 0. For the inductive step we prove that for each summand in P there is a provably equivalent summand in Q and vice versa. The theorem then follows by **S2**–**S4**.

Take a summand $\overline{a}x.\, P'$ of P; we have $P \xrightarrow{\overline{a}x} P'$, so $Q \xrightarrow{\overline{a}x} Q'$ with $P' \dot\sim_{\mathrm{L}} Q'$. Then $\overline{a}x.\, Q'$ is a summand in Q. Moreover, by induction and **IP1** we get $\mathcal{LB} \vdash \overline{a}x.\, P' = \overline{a}x.\, Q'$.

For the input summands first apply **A** so that all input actions get the same object, say x, distinct from any name in $\mathrm{fn}(P, Q)$. Assume $a(x).\, P'$ is a summand in P. Then $P \xrightarrow{a(x)} P'$. Since $P \dot\sim_{\mathrm{L}} Q$ also $Q \xrightarrow{a(x)} Q'$ s.t. $P'\{y/x\} \dot\sim_{\mathrm{L}} Q'\{y/x\}$ for all y. Thus, $a(x).\, Q'$ must be a summand in Q. By induction $\mathcal{LB} \vdash P'\{y/x\} = Q'\{y/x\}$ for any fixed y. So, through a finite proof involving **IP2** (note that $\mathrm{fn}(P)$ is finite) we have $\mathcal{LB} \vdash a(x).\, P' = a(x).\, Q'$.

Conversely, for every summand in Q there is a provably equivalent summand in P by a symmetric argument. □

3.3 Early Bisimilarity

Consider the law:

$$\boxed{\textbf{SP} \quad a(x).\, P + a(x).\, Q \;=\; a(x).\, P + a(x).\, Q + a(x).\, \Big([x = y]\; P\; Q\Big)}$$

The axiom system $\mathcal{EB}$ we propose for early bisimilarity is $\mathcal{LB} \cup \{\textbf{SP}\}$. The new axiom says that when there is a choice between two input prefixes on the same port, say $a(x).\, P$ and $a(x).\, Q$, a third alternative $a(x).\, \Big([x = y]\; P\; Q\Big)$ can be added. The latter behaves as one of the two original prefixes; the value received on a determines which one.

Proposition 3.4 (soundness of $\mathcal{EB}$) *If $\mathcal{EB} \vdash P = Q$ then $P \dot\sim_{\mathrm{E}} Q$.*

Proof: Since $\dot\sim_{\mathrm{L}}$ is included in $\dot\sim_{\mathrm{E}}$ we only need to prove that the new law **SP** and the inference laws **IS** and **IP** are sound. It is easy to establish the necessary early bisimulation. □

The completeness result uses the following additional concepts. Let a be a port name and P a hnf. Then P_a is the sum of all summands in P of type $a(x).\,P'$, i.e., all summands which are input prefixes on the port a. The *output part* of P, written P_{out}, is the sum of all output prefix summands in P. So through **S3** and **S4**, P can be written

$$\mathcal{EB} \vdash P = \sum_{a \in A} P_a \; + \; P_{\text{out}} \tag{5}$$

for some suitable set of port names A.

Theorem 3.5 (completeness of $\mathcal{EB}$ for $\dot{\sim}_{\text{E}}$) *If $P \dot{\sim}_{\text{E}} Q$ then $\mathcal{EB} \vdash P = Q$.*

Proof: Again it suffices to establish the proof for hnf's P and Q, and the proof is by induction on depth. The base case is trivial. For the inductive step we will prove that if $P \dot{\sim}_{\text{E}} Q$ then, for any a, $\mathcal{EB} \vdash P_a = Q_a$. Furthermore, an argument similar to that in the proof of Theorem 3.3 establishes that each output prefix in P has a provably equivalent output prefix in Q. Thus $\mathcal{EB} \vdash P_{\text{out}} = Q_{\text{out}}$. So by (5):

$$\begin{aligned} \mathcal{EB} \vdash P &= \textstyle\sum_{a \in A} P_a + P_{\text{out}} \\ &= \textstyle\sum_{a \in A} Q_a + Q_{\text{out}} \\ &= Q \end{aligned}$$

We will now show that $\mathcal{EB} \vdash P_a = Q_a$. Choose an $x \notin \text{fn}(P, Q)$ and apply **A** so that all top-level prefixes in P_a and Q_a use the same object name x. That is, we get

$$\mathcal{EB} \vdash P_a = \sum_{i=1}^{n} a(x).\,P_i, \qquad \mathcal{EB} \vdash Q_a = \sum_{j=1}^{m} a(x).\,Q_j \tag{6}$$

To establish that P_a and Q_a are provably equivalent we will "saturate" these agents by adding summands to them. The new summands will be constructed by pieces of P_a and Q_a. We will prove that adding such summands does not affect early bisimilarity, and that we can add enough of them to make P_a and Q_a provably equivalent. The key to the proof is to exhibit, for each $i \in [1, n]$ an agent R_i with the following properties:

$$\mathcal{EB} \vdash a(x).\,P_i = a(x).\,R_i \tag{7}$$

$$\mathcal{EB} \vdash Q_a = Q_a + a(x).\,R_i \tag{8}$$

Repeated application of (8) for $i = 1, \ldots, n$ gives

$$\mathcal{EB} \vdash Q_a = Q_a + \sum_{i=1}^{n} a(x).\,R_i$$

But by (7) and (6) we can write this as

$$\mathcal{EB} \vdash Q_a = Q_a + P_a$$

In a completely symmetric way (just exchange P and Q) we also derive

$$\mathcal{EB} \vdash P_a = P_a + Q_a$$

So, by **S2**, we get that both P_a and Q_a are provably equivalent to the same agent, $P_a + Q_a$. It follows that

$$\mathcal{EB} \vdash P_a = Q_a$$

as required.

To complete the proof we now need to define R_i and derive the crucial properties (7) and (8). First, it is trivial to construct an early bisimulation containing (P_a, Q_a) as a subset of an early bisimulation containing (P, Q). Thus we get

$$P_a \stackrel{\cdot}{\sim}_{\mathrm{E}} Q_a \tag{9}$$

Consider a transition from P_a:

$$P_a \xrightarrow{a(x)} P_i$$

Let y be any object name. By (9) and Definition 2.4 there must be a matching transition arising from a summand, call it j, in Q_a:

$$Q_a \xrightarrow{a(x)} Q_j$$

s.t.

$$P_i\{y/x\} \stackrel{\cdot}{\sim}_{\mathrm{E}} Q_j\{y/x\} \tag{10}$$

There may be several indices j satisfying this but there must be at least one. For each $i \in [1,n]$ and each object name y, designate such an index $J(i,y) \in [1,m]$; thus J is a function from $[1,n]$ and object names to $[1,m]$ satisfying $P_i\{y/x\} \stackrel{\cdot}{\sim}_{\mathrm{E}} Q_{J(i,y)}\{y/x\}$. Observe that J is not necessarily an injection.

By induction (P_i and Q_j have smaller depths than P and Q respectively), we get for any i and y:

$$\mathcal{EB} \vdash P_i\{y/x\} = Q_{J(i,y)}\{y/x\} \tag{11}$$

Let $y_1, \ldots, y_k$ be all the free object names in P_a and Q_a. If there are no such names then let $k = 0$. Let i be any index in $[1,n]$. Define the agents $S_{i,l}$ for $l \in [0,k]$ by

$$S_{i,0} \equiv Q_{J(i,x)} \tag{12}$$

$$S_{i,l} \equiv [x = y_l]\, Q_{J(i,y_l)}\, S_{i,l-1} \qquad \text{for } l \in [1,k] \tag{13}$$

Therefore, when x is instantiated to $z \in \{x, y_1, \ldots, y_l\}$, $S_{i,l}$ reduces to $Q_{J(i,z)}$ by successive evaluations of the outermost condition. Thus a simple induction on $l \in [0,k]$ establishes that:

$$\text{For all } z \in \{x, y_1, \ldots, y_l\}\text{:} \quad \mathcal{EB} \vdash Q_{J(i,z)}\{z/x\} = S_{i,l}\{z/x\} \tag{14}$$

Now define

$$R_i \equiv S_{i,k}$$

thus, we are ready to prove (7) and (8). Consider first (7). From (14) we have, for any $z \in \{x, y_1, \ldots, y_k\}$,

$$\mathcal{EB} \vdash Q_{J(i,z)}\{z/x\} = R_i\{z/x\}$$

Apply (11) to the left hand side of this to get

$$\mathcal{EB} \vdash P_i\{z/x\} = R_i\{z/x\} \tag{15}$$

Since $\{x, y_1, \ldots, y_k\} = \mathrm{fn}(P_a, Q_a, x) \supseteq \mathrm{fn}(P_i, Q_i, x)$ we can apply **IP2** to obtain

$$\mathcal{EB} \vdash a(x).\,P_i = a(x).\,R_i$$

which is just (7).

Finally (8) follows by a similar induction. We will prove that for all $l \in [0, k]$:

$$\mathcal{EB} \vdash Q_a = Q_a + a(x).\,S_{i,l} \tag{16}$$

The base case ($l = 0$) is, by (12),

$$\mathcal{EB} \vdash Q_a = Q_a + a(x).\,Q_{J(i,x)}$$

But this is immediate from **S2**–**S4** since $a(x).\,Q_{J(i,x)}$ is a summand in Q_a. For the inductive step let $0 < l \leq k$. By induction we may assume

$$\mathcal{EB} \vdash Q_a = Q_a + a(x).\,S_{i,l-1}$$

But $a(x).\,Q_{J(i,y_l)}$ is a summand in Q_a, so we get by **S2**–**S4**:

$$\mathcal{EB} \vdash Q_a = Q_a + a(x).\,Q_{J(i,y_l)} + a(x).\,S_{i,l-1} \tag{17}$$

Now for the first (and only!) time we apply **SP**, to the two rightmost summands:

$$\mathcal{EB} \vdash Q_a = Q_a + a(x).\,Q_{J(i,y_l)} + a(x).\,S_{i,l-1} + a(x).\,\big([x = y_l]\; Q_{J(i,y_l)}\; S_{i,l-1}\big)$$

Apply (17) to the three first terms after the "=" to get

$$\mathcal{EB} \vdash Q_a = Q_a + a(x).\,\big([x = y_l]\; Q_{J(i,y_l)}\; S_{i,l-1}\big)$$

and by (13) this is just (16). This completes the inductive step and the proof of (8), and hence also the proof of the theorem. □

4 Late and Early Congruence

4.1 Substitutions and Conditions

Below, let $\sim$ stand for either of $\sim_{\mathrm{L}}$ or $\sim_{\mathrm{E}}$ and similarly let $\dot{\sim}$ stand for either of $\dot{\sim}_{\mathrm{L}}$ or $\dot{\sim}_{\mathrm{E}}$. Following Definition 2.5, an immediate solution to the axiomatisation of $\sim$ is to take the system for $\dot{\sim}$ and add the inference rule:

> if $\mathcal{LB}$ (resp. $\mathcal{EB}$) $\vdash P\sigma = Q\sigma$ for all substitutions σ, then infer $P = Q$

But this would not help in avoiding the case analysis as mentioned in Section 1, nor would it give an independent characterisation of the relation $\sim$, since it refers to another equivalence and its axiom system. Our contribution here is to present an independent characterisation which avoids the case analysis problem. Before proceeding to the axiom system and completeness proof we will need some auxiliary definitions and results about substitutions and conditions. These are motivated by a difficulty with the completeness proof outlined below.

Let $\asymp$ be some behavioural equivalence, and $\mathcal{AS}$ an axiom system for $\asymp$ and consider the following standard argument for proving that any two $\asymp$-equivalent hnf's P and Q are provably equal from $\mathcal{AS}$. For any summand $\alpha.P'$ of P, the definition of $\asymp$ yields that there exists a summand $\beta.Q'$ of Q s.t. $\mathcal{AS} \vdash \beta.Q = \alpha.Q$ and $P' \asymp Q'$. From this it follows $\mathcal{AS} \vdash P = Q$ by induction on the depth of the hnf's. Unfortunately, this last step does not quite work for $\sim$. Since $\sim$ is defined in terms of $\dot{\sim}$, we only can derive $P' \dot{\sim} Q'$ (or $P'\sigma \dot{\sim} Q'\sigma$ for some substitution σ), and in general this does not imply $P' \sim Q'$. To overcome this problem we shall appeal to Lemma 4.5 below, which allows us to lift $\dot{\sim}$ up to $\sim$ on agents of a special format. Indeed, the hnf's will be defined to comply to the format required by this lemma.

We write $\varphi \Rightarrow \psi$ if for each substitution σ it holds that $[\![\varphi\sigma]\!]$ implies $[\![\psi\sigma]\!]$ (that is, whenever φ is true, then ψ is true); further, $\varphi \Leftrightarrow \psi$ if both $\varphi \Rightarrow \psi$ and $\psi \Rightarrow \varphi$ hold (that is, φ and ψ are semantically equivalent).

Definition 4.1 *A condition φ is* consistent *if it is satisfiable, i.e. for some σ, $[\![\varphi\sigma]\!] =$ True. Let V be a set of names; a condition φ is* complete on V *if for some equivalence relation $\mathcal{R}$ on V, called the equivalence relation* corresponding to φ, *it holds*

$$\varphi \Rightarrow [x = y] \text{ iff } x\mathcal{R}y, \text{ and } \varphi \Rightarrow [x \neq y] \text{ iff } \neg(x\mathcal{R}y)$$

For instance, $\varphi = [(a = b) \wedge (b = c) \wedge (c \neq d)]$ is complete on $V = \{a, b, c, d\}$, and $\mathcal{R}$ with the equivalence classes $\{\{a, b, c\}, \{d\}\}$ is the equivalence relation corresponding to φ. In general, an equivalence relation on a set of names decrees that some names are equivalent and some are not; the corresponding complete condition implies all those equalities and inequalities among names. Thus a condition φ complete on a set V of names is *maximally consistent* in the sense that making φ stronger, by conjoining a condition on names in V not implied by φ, makes φ inconsistent.

Definition 4.2 *A substitution* σ agrees *with a condition* φ*, and* φ agrees *with* σ*, if for all* x, y *which appear in* φ *it holds that* $\sigma(x) = \sigma(y)$ *iff* $\varphi \Rightarrow [x = y]$.

Lemma 4.3 *Let* V *be a set of names and let* φ *be complete on* V.

1. *If* σ *and* σ' *are substitutions on* V *which both agree with* φ*, then* $\sigma = \sigma'\rho$ *for some injective substitution* ρ.
2. *If* ψ *is another condition with names in* V*, then either* $\varphi \wedge \psi$ *is unsatisfiable or* $\varphi \wedge \psi \Leftrightarrow \varphi$.
3. *If* ψ *is another condition complete on* V *s.t.* φ *and* ψ *agree with the same substitution* σ *then* $\varphi \Leftrightarrow \psi$.

PROOF:

1. Let R be the equivalence corresponding to φ. Any substitution σ agreeing with φ must define one "σ-representative" from each equivalence class in R and send all elements in V to the σ-representative of its class, so ρ is simply the injection which maps the σ'-representatives to the σ-representatives.
2. Follows from the fact that φ is maximally consistent.
3. By 2 above, both $\varphi \wedge \psi \Leftrightarrow \varphi$ and $\varphi \wedge \psi \Leftrightarrow \psi$.

$\square$

Lemma 4.4 *Suppose* $P \stackrel{\cdot}{\sim} Q$ *and* σ *is injective on* $\mathrm{fn}(P, Q)$*. Then also* $P\sigma \stackrel{\cdot}{\sim} Q\sigma$.

PROOF: Omitted; see [MPW92]. $\square$

Lemma 4.5 below reveals an interesting relationship between conditionals and substitutions: The outermost conditionals of two agents can be exploited to restrict the quantification over substitutions in the definition of $\sim$. In particular, when the conditions are complete on the free names of the agents, one substitution is enough. In the lemma we use $V - V'$ for the set difference between V and V', i.e. $\{a \,:\, a \in V \text{ and } a \notin V'\}$. Moreover, we let the *proper domain* of a substitution σ, written $\mathrm{prdom}(\sigma)$, be the set of names on which σ does not act as an identity, i.e. $\mathrm{prdom}(\sigma) = \{a \ : \ \sigma(a) \neq a\}$; similarly, we let the *proper codomain* of σ be the image of the proper domain, i.e. $\mathrm{prcod}(\sigma) = \{a \ : \ \text{for some } b,\ b \neq \sigma(b) = a\}$.

Lemma 4.5 *Let* $P \equiv \varphi\, P'$ *and* $Q \equiv \varphi\, Q'$*, with* φ *complete on a set* V_1 *of names. Define* $V_2 = \mathrm{fn}(P, Q) - V_1$ *and suppose that* σ_1 *is a substitution s.t.:*

1. $\mathrm{prdom}(\sigma_1) \subseteq V_1$ *and* σ_1 *agrees with* φ*;*
2. $\mathrm{prcod}(\sigma_1) \cap V_2 = \emptyset$ *;*
3. *for all* σ_2 *with* $\mathrm{prdom}(\sigma_2) \subseteq V_2$*, it holds that* $P\sigma_1\sigma_2 \mathrel{\dot{\sim}} Q\sigma_1\sigma_2$*.*

Then $P \sim Q$*.*

Proof: If W is a set of names, we write $W\sigma$ for the set $\{\sigma(a) \ : \ a \in W\}$. Moreover, we write $\sigma_{|W}$ for the restriction of σ to W; thus, $\sigma_{|W}(a)$ is $\sigma(a)$ if $a \in W$, otherwise $\sigma(a)$ is undefined.

We have to show that for each σ, $P\sigma \mathrel{\dot{\sim}} Q\sigma$. It is enough to consider substitutions σ whose proper domain is contained in $\mathrm{fn}(P, Q)$. If σ does not agree with φ, then $[\![\varphi\sigma]\!] = \mathit{False}$, therefore $P\sigma \mathrel{\dot{\sim}} Q\sigma \mathrel{\dot{\sim}} \mathbf{0}$. The case when σ agrees with φ is more delicate. We first show the following auxiliary fact for σ; here W is $V_1 \cup V_2$ and σ_1 is the substitution in the assertion of the lemma.

> *There are a substitution* σ_2 *with* $\mathrm{prdom}(\sigma_2) \subseteq V_2$ *and a substitution* ρ *injective on* $W\sigma_1\sigma_2$ *s.t.* $\sigma_{|W} = (\sigma_1\sigma_2\rho)_{|W}$*.*

To prove this, we proceed by induction on the cardinality of V_2. When V_2 is empty, the result follows from Lemma 4.3(1). Suppose V_2 is made of $n + 1$ names, and let $a \in V_2$, $W^a = W - \{a\}$. By the induction hypothesis, there are σ_2^a and ρ^a satisfying certain properties and with

$$\sigma_{|W^a} = (\sigma_1\sigma_2^a\rho^a)_{|W^a} \tag{18}$$

Now we distinguish two cases; we write $\sigma[{}^x\!/\!_y]$ for the substitution which maps y into x and behaves like σ elsewhere.

1. There is $b \in W^a\sigma_1\sigma_2^a$ s.t. $\rho^a(b) = \sigma(a)$. Then take

$$\sigma_2 = \sigma_2^a[{}^b\!/\!_a] \qquad\qquad \rho = \rho^a$$

   The substitution ρ is injective on $W\sigma_1\sigma_2 = W^a\sigma_1\sigma_2^a \cup \{\sigma_2(a)\} = W^a\sigma_1\sigma_2^a \cup \{b\} = W^a\sigma_1\sigma_2^a$ because by induction ρ^a is injective on $W^a\sigma_1\sigma_2^a$.

2. There is no $b \in W^a\sigma_1\sigma_2^a$ s.t. $\rho^a(b) = \sigma(a)$. Let c be a fresh name ($c \notin W^a\sigma_1\sigma_2^a$) and take
$$\sigma_2 = \sigma_2^a[c/a] \qquad \rho = \rho^a[\sigma(a)/c]$$
The substitution ρ is injective on $W\sigma_1\sigma_2 = W^a\sigma_1\sigma_2^a \cup \{c\}$: Take two names b_1 and $b_2 \in W\sigma_1\sigma_2$: If $c \neq b_1, b_2$, then $\rho(b_1) = \rho^a(b_1) \neq \rho^a(b_2) = \rho(b_2)$ by induction. Otherwise, if one of them, say b_2, is c, then $\rho(b_1) = \rho^a(b_1) \neq \sigma(a) = \rho(c)$ by the assumption made on a.

From the definition of $\sigma_1\sigma_2\rho$ on a and (18), we have $\sigma_{|W} = (\sigma_1\sigma_2\rho)_{|W}$, as requested. Note that in both above cases the decomposition of σ is made possible by the hypothesis that $\mathrm{prdom}(\sigma_1) \subseteq V_1$ and $\mathrm{prcod}(\sigma_1) \cap V_2 = \emptyset$; in particular this insures that $V_2\sigma_1 = V_2$ and that σ_1 maps no name in V_1 to a. Using this decomposition of σ, we have

$$P\sigma \equiv P\sigma_1\sigma_2\rho \qquad Q\sigma \equiv Q\sigma_1\sigma_2\rho \tag{19}$$

Now we can prove $P\sigma \mathrel{\dot{\sim}} Q\sigma$: From the hypothesis (3) of the lemma, $P\sigma_1\sigma_2 \mathrel{\dot{\sim}} Q\sigma_1\sigma_2$; since ρ is injective on $\mathrm{fn}(P\sigma_1\sigma_2, Q\sigma_1\sigma_2)$, by Lemma 4.4 also

$$P\sigma_1\sigma_2\rho \mathrel{\dot{\sim}} Q\sigma_1\sigma_2\rho$$

from which, by (19), $P\sigma \mathrel{\dot{\sim}} Q\sigma$. □

As an example, let φ be $[(a = b) \wedge (b \neq c)]$. Suppose we want to know whether $\varphi\, P \;\sim\; \varphi\, Q$ holds, where $\mathrm{fn}(P, Q) = \{a, b, c, d\}$. Since φ is complete on $\{a, b, c\}$, Lemma 4.5 tells us that it is enough to take a substitution which agrees with φ, like $\{b/a\}$, and check the bisimilarity of $\varphi\, P\{b/a\}$ and $\varphi\, Q\{b/a\}$ for all instantiations of the name d; that is

$$\varphi\, P \;\sim\; \varphi\, Q \quad \text{iff} \quad \forall\, x\; (\varphi\, P)\{b/a\}\{x/d\} \mathrel{\dot{\sim}} (\varphi\, Q)\{b/a\}\{x/d\}$$

4.2 Late Congruence

In the system $\mathcal{LB}$ for late bisimilarity, we used **C1** and **C2** to eliminate conditionals. For the congruence these axioms are not sound, since a non-injective substitution σ may make $[\![\varphi\sigma]\!]$ different from $[\![\varphi]\!]$. We can obtain sound laws by strengthening the side conditions of **C1** and **C2** to $\varphi \Leftrightarrow \mathit{True}$ and $\varphi \Leftrightarrow \mathit{False}$ respectively, but then the system would be incomplete since we would only be able to eliminate conditionals whose conditions are semantically equivalent to *True* or *False*. So instead of eliminating conditionals we add a few axioms for manipulating them.

The axiom system $\mathcal{LC}$ for late congruence is given in Table 2. Each axiom has a specific function, this will give us simple independence proofs in Section 6. Note that

IP	If $P = Q$ then $\alpha.P = \alpha.Q$	
IS	If $P = Q$ then $P + R = Q + R$	
IC	If $P = Q$ then $\varphi\, P = \varphi\, Q$	
S1	$P + \mathbf{0} = P$	
S2	$P + P = P$	
S3	$P + Q = Q + P$	
S4	$P + (Q + R) = (P + Q) + R$	
C3	if $\varphi \Leftrightarrow \psi$ then $\varphi\, P = \psi\, P$	
C4	$\mathit{False}\; P = \mathit{False}\; Q$	
C5	$\varphi\; P\; P = P$	
C6	$\varphi\; P\; Q = \neg\varphi\; Q\; P$	
CC1	$\varphi\; (\psi\; P) = [\varphi \wedge \psi]\; P$	
SC1$*$	$\varphi\,(P_1 + P_2)\,(Q_1 + Q_2) = \varphi\, P_1\, Q_1 + \varphi\, P_2\, Q_2$	
CP1	$\varphi\,(\alpha.P) = \varphi\,(\alpha.\,\varphi\, P)$	if $\mathrm{bn}(\alpha) \cap \mathrm{n}(\varphi) = \emptyset$
CP2	$[x = y]\,\alpha.P = [x = y]\,(\alpha\{x/y\}).P$	

Table 2: Axiom system $\mathcal{LC}$ for late congruence.

there is an interesting correspondence between the axioms **S1**-**S4** for sum, and the axioms **C4**, **C5**, **C6** and **CC1** for conditionals: **S1** and **C4** say, respectively, that there is a special agent **0** and a special condition *False*; **S2** and **C5** imply idempotence; **S3** and **C6** imply commutativity; **S4** and **CC1** deal with the nesting of operators. We preferred **C4** to the similar **C4**$'$: *False* $Q = \mathbf{0}$ because we regard the former as slightly simpler since it cannot introduce a conditional. This comes up in the independence proofs of Section 6 and in the inspection of the inter-derivability of the axioms. **C4**$'$ alone gives **C4**:

$$\mathit{False}\ Q \overset{\mathbf{C4}'}{=} \mathbf{0} \overset{\mathbf{C4}'}{=} \mathit{False}\ P$$

whereas, for the converse, also **C5** is needed:

$$\mathit{False}\ Q \overset{\mathbf{C4}}{=} \mathit{False}\ \mathbf{0} \overset{\mathbf{C5}}{=} \mathbf{0}$$

However, the replacement of **C4** with **C4**$'$ would not affect the independence of our axioms (see Remark 6.4).

The axiom **C3** is expressed in terms of semantical equality between conditions. We chose this, as opposed to a set of axioms for syntactic transformations of conditions, to gain clarity and to focus on the more interesting issue of the interaction between conditionals and the rest of the language.

The axioms **SC1**$*$ and **CP1** say how conditionals distribute over sum and prefix. Note that **C5**, **C6** and **SC1**$*$ are the only axioms involving binary conditionals. Finally, the axiom **CP2** incorporates the meaning of the matching construct; underneath it the matched names can be regarded as equal and are hence interchangeable. From **CP2** and our distributive laws we can infer the more powerful

$$[x=y]\,P \;=\; [x=y]\,P\{y/x\}$$

where the substitution acts on the whole operand of the conditional.

Proposition 4.6 (soundness of $\mathcal{LC}$) *If* $\mathcal{LC} \vdash P = Q$ *then* $P \sim_{\mathrm{L}} Q$.

PROOF: By establishing appropriate bisimulations; all cases are simple. □

The remainder of the section is devoted to the proof of the completeness of $\mathcal{LC}$. First some useful derived laws:

$$\begin{array}{ll}
\mathbf{SC1} & \varphi\,(P+Q) = \varphi\,P + \varphi\,Q \\
\mathbf{SC2} & \varphi\,P\;Q = \varphi\,P + \neg\varphi\,Q \\
\mathbf{C7} & \mathit{True}\;P\;Q = P \\
\mathbf{CC2} & [\varphi \vee \psi]\,P = \varphi\;P\,(\,\psi\,P) \\
\mathbf{SC3} & P = \varphi\,P + \neg\varphi\,P \\
\mathbf{SC4} & [\varphi \vee \psi]\;P = \varphi\;P + \psi\;P
\end{array}$$

Proof: We show the derivation of each of them in order:

SC1: $\varphi\,(P+Q) \overset{\mathbf{S1}}{=} \varphi\,(P+Q)\,(\mathbf{0}+\mathbf{0}) \overset{\mathbf{SC1}*}{=} \varphi\,P + \varphi\,Q$

SC2: $\varphi\,P\;Q \overset{\mathbf{S1},\mathbf{S3}}{=} \varphi\,(P+\mathbf{0})\,(\mathbf{0}+Q) \overset{\mathbf{SC1}*}{=} \varphi\,P + \varphi\,\mathbf{0}\;Q \overset{\mathbf{C6}}{=} \varphi\,P + \neg\varphi\,Q$

C7: $P \overset{\mathbf{C5}}{=} \mathit{True}\;P\;P \overset{\mathbf{SC2}}{=} \mathit{True}\;P + \mathit{False}\;P \overset{\mathbf{C4}}{=}$
$\mathit{True}\;P + \mathit{False}\;Q \overset{\mathbf{SC2}}{=} \mathit{True}\;P\;Q$

CC2: $[\varphi \vee \psi]\,P \overset{\mathbf{C5}}{=} [\varphi \vee \psi]\,(\,\varphi\;P\;P\,) \overset{\mathbf{SC2}}{=} [\varphi \vee \psi]\,(\,\varphi\,P + \neg\varphi\,P) \overset{\mathbf{SC1},\mathbf{CC1}}{=}$
$[(\varphi \vee \psi) \wedge \varphi]\,P + [(\varphi \vee \psi) \wedge \neg\varphi]\,P \overset{\mathbf{C3}}{=} \varphi\,P + [\neg\varphi \wedge \psi]\,P \overset{\mathbf{CC1}}{=}$
$\varphi\,P + \neg\varphi\,(\,\psi\,P) \overset{\mathbf{SC2}}{=} \varphi\;P\,(\,\psi\,P).$

SC3: $P \overset{\mathbf{C5}}{=} \varphi\;P\;P \overset{\mathbf{SC2}}{=} \varphi\,P + \neg\varphi\,P.$

SC4: $[\varphi \vee \psi]\,P \overset{\mathbf{CC2}}{=} \varphi\;P\,(\,\psi\,P) \overset{\mathbf{SC2},\mathbf{CC1}}{=} \varphi\,P + [\neg\varphi \wedge \psi]\,P \overset{\mathbf{SC3},\mathbf{CC1}}{=}$
$[\psi \wedge \varphi]\,P + [\neg\psi \wedge \varphi]\,P + [\psi \wedge \neg\varphi]\,P \overset{\mathbf{S2},\mathbf{S3},\mathbf{S4}}{=}$
$[\psi \wedge \varphi]\,P + [\neg\psi \wedge \varphi]\,P + [\psi \wedge \varphi]\,P + [\psi \wedge \neg\varphi]\,P \overset{\mathbf{C3},\mathbf{CC1}}{=}$
$\psi\,(\,\varphi\,P) + \neg\psi\,(\,\varphi\,P) + \varphi\,(\,\psi\,P) + \neg\varphi\,(\,\psi\,P) \overset{\mathbf{SC3}}{=} \varphi\,P + \psi\,P$

□

Note that through **SC1** it is not possible to derive the binary **SC1∗**. Intuitively, we need **SC1∗** to split a binary conditional $\varphi\;P\;Q$ where φ is a non-trivial condition and P and Q have non-zero and different depths, as examplified by:

$$[a\!=\!b]\;(a(x).\,\mathbf{0})\;\;(a(x).\,b(x).\,\mathbf{0}) \overset{\mathbf{SC2}}{=} [a\!=\!b]\,a(x).\,\mathbf{0} + [a\!\neq\!b]\,a(x).\,b(x).\,\mathbf{0}$$

This instance of **SC2** cannot be derived from **SC1**.

The key to the completeness proofs in this and in the next section is the "saturation" of conditions, i.e. they are made complete w.r.t. a set V of names. Such a set is present as a parameter in the definition of head normal forms. In the completeness proofs, the hnf's to which two agents under investigation are reduced will depend upon the free names in both agents.

Definition 4.7 (head normal forms) *Let V be a set of names. We say that P is in* head normal form *on V if P is of the form*

$$\textstyle\sum_{i\in I}\ \varphi_i\ \alpha_i.\,\varphi_i P_i$$

where for all i,

1. $\mathrm{bn}(\alpha_i) \notin V$*;*
2. φ_i *is complete on V.*

Note that each condition φ_i occurs twice, before and after α_i.

Lemma 4.8 *For each agent P, and for each finite set of names V with* $\mathrm{fn}(P) \subseteq V$*, there is an agent H of no greater depth than P and in hnf on V, s.t.* $\mathcal{LC} \vdash P = H$.

Proof: By structural induction over agents. $\mathbf{0}$ is already in hnf. The inductive step has three cases. The first is when $P \equiv Q + R$ and just requires the induction hypothesis.

The second case is when $P \equiv \alpha.\,P'$. Let $\mathcal{R}_1, \ldots, \mathcal{R}_n$ enumerate all possible equivalence relations on V; if φ_i is a complete condition corresponding to $\mathcal{R}_i$, then

$$\bigvee_{i=1}^{n} \varphi_i \Leftrightarrow \mathit{True} \tag{20}$$

since any σ must agree with at least one of the φ_i on V. Using this, and alpha-conversion to ensure that a name bound by α (if any) is not in V, we get

$$P \overset{\mathbf{C7}}{=} \mathit{True}\ P \overset{\mathbf{C3}}{=} [\bigvee_{i=1}^{n} \varphi_i]\ P \overset{\mathbf{SC4}}{=} \sum_{i=1}^{n} \varphi_i\ P \equiv \sum_{i=1}^{n} \varphi_i\ \alpha.\,P' \overset{\mathbf{CP1}}{=} \sum_{i=1}^{n} \varphi_i\ \alpha.\ \varphi_i\ P'$$

where the last agent is in hnf on V.

The third case is when $P \equiv \varphi\ Q\ R$. The axiom **SC2** gives $\mathcal{LC} \vdash P = \varphi\, Q + \neg\varphi\ R$. By induction Q and R can be put into hnf on V:

$$\mathcal{LC} \vdash Q = \sum_{i=1}^{n} \varphi_i\ \alpha_i.\ \varphi_i\ Q_i\ ; \quad R = \sum_{j=1}^{m} \varphi_j\ \alpha_j.\ \varphi_j\ R_j$$

which gives

$$\mathcal{LC} \vdash P = \varphi \sum_{i=1}^{n} \varphi_i\ \alpha_i.\ \varphi_i\ Q_i\ +\ \neg\varphi \sum_{j=1}^{m} \varphi_j\ \alpha_j.\ \varphi_j\ R_j$$

If n or m is $\emptyset$ then with **C5** (precisely, the special case of **C5**, $\varphi\ \mathbf{0}\ \mathbf{0}\ = \mathbf{0}$) and **S1** the corresponding summand can be eliminated. Otherwise, distributing the condition over the sum with **SC1** and applying **CC1**, we get

$$\mathcal{LC} \vdash P = \sum_{i=1}^{n} [\varphi \wedge \varphi_i]\ \alpha_i.\ \varphi_i\ Q_i\ +\ \sum_{j=1}^{m} [\neg\varphi \wedge \varphi_j]\ \alpha_j.\ \varphi_j\ R_j$$

Similarly, using **CP1** and **CC1**, we get

$$\mathcal{LC} \vdash P = \sum_{i=1}^{n} [\varphi \wedge \varphi_i]\, \alpha_i .\, [\varphi \wedge \varphi_i]\, Q_i \;+\; \sum_{j=1}^{m} [\neg\varphi \wedge \varphi_j]\, \alpha_j .\, [\neg\varphi \wedge \varphi_j]\, R_j$$

By Lemma 4.3(2), each $[\varphi \wedge \varphi_i]$ (resp. $[\neg\varphi \wedge \varphi_j]$) is semantically equivalent either to *False* or to φ_i (resp. φ_j): Hence from **C3**, **C6**, **C7** and **S1** each summand of P can either be removed or put into the form $\varphi_i\,(\alpha_i .\, \varphi_i\, Q_i)$ (resp. $\varphi_j\,(\alpha_j .\, \varphi_j\, R_j)$), which is exactly the form of the summands of a hnf. $\square$

Theorem 4.9 (completeness of $\mathcal{LC}$ for $\sim_{\mathrm{L}}$) *If $P \sim_{\mathrm{L}} Q$, then $\mathcal{LC} \vdash P = Q$.*

PROOF: By Lemma 4.8, it is enough to prove the assertion when P, Q are in hnf on $\mathrm{fn}(P, Q)$. As for $\mathcal{LB}$ the proof is by induction on the depth of $P + Q$ and shows that each summand of P is provably equal to some summand of Q. Let $\varphi\, \alpha .\, P'$ be a summand of P and σ a substitution which agrees with φ; we can also assume that σ acts as identity on the names not free in P or Q. We have $P\sigma \xrightarrow{\alpha\sigma} P'\sigma$. By definition of $\sim_{\mathrm{L}}$, it holds that $P\sigma \mathrel{\dot{\sim}_{\mathrm{L}}} Q\sigma$. Let $\psi\, \beta .\, Q'$ be the summand of Q used to simulate the transition from $P\sigma$. Using alpha-conversion we can assume that the bound names (if any) in α and β are the same. From this, and the definition of $\dot{\sim}_{\mathrm{L}}$, we have:

(a) ψ agrees with σ,

(b) $\beta\sigma = \alpha\sigma$,

(c) if α, β are outputs, then $P'\sigma \mathrel{\dot{\sim}_{\mathrm{L}}} Q'\sigma$,

(d) if α, β are inputs with bound name x, then $P'\sigma\{z/x\} \mathrel{\dot{\sim}_{\mathrm{L}}} Q'\sigma\{z/x\}$, for all z.

The conditions φ and ψ are complete on $\mathrm{fn}(P, Q)$ and agree with σ. By Lemma 4.3(3), $\varphi \Leftrightarrow \psi$, hence by **C3**, $\mathcal{LC} \vdash \psi\, \beta .\, Q' = \varphi\, \beta .\, Q'$. We would also like to infer $\mathcal{LC} \vdash \varphi\, \beta .\, Q' = \varphi\, \alpha .\, Q'$. We can do so exploiting **CP2**. To see this, suppose that α and β differ in the names a and b, that is, $\beta\{a/b\} = \alpha\{a/b\}$. Since $\alpha\sigma = \beta\sigma$, it must be that $\sigma(a) = \sigma(b)$. But φ agrees with σ; hence $\varphi \Rightarrow [a = b]$. Therefore we get

$$\begin{array}{l} \varphi\, \beta .\, Q' \overset{\mathbf{C3,CC1}}{=} \varphi\,(\,[a = b]\, \beta .\, Q') \overset{\mathbf{CP2}}{=} \varphi\big(\,[a = b]\,(\beta\{a/b\}) .\, Q'\big) \equiv \\ \varphi\big(\,[a = b]\,(\alpha\{a/b\}) .\, Q'\big) \overset{\mathbf{CP2,\,CC1,\,C3}}{=} \varphi\, \alpha .\, Q' \end{array}$$

It remains to prove that $P' \sim_{\mathrm{L}} Q'$; this would allow us to use the induction hypothesis to infer $\mathcal{LC} \vdash P' = Q'$ and thus conclude that $\mathcal{LC} \vdash \psi\, \beta .\, Q' = \varphi\, \alpha .\, P'$. By definition of hnf, P' and Q' are of the form $\varphi\, P''$ and $\psi\, Q'' \overset{\mathbf{C3}}{=} \varphi\, Q''$. Now, $\varphi\, P'' \sim_{\mathrm{L}} \varphi\, Q''$

can be derived from Lemma 4.5. In the premises of the lemma, take $V_1 = \mathrm{fn}(P, Q)$ and $\sigma_1 = \sigma$. By assumption, σ is complete on φ and the proper domain of σ is contained in $\mathrm{fn}(P, Q)$, so condition (1) of the lemma is satisfied. For conditions (2) and (3), we distinguish the case when α is an output or an input: In the former, we have $V_2 = \emptyset$ and then use clause (c) above; in the latter, if x is the bound name, we have $V_2 = \{x\}$ and then use (d) together with alpha-conversion to guarantee that x is fresh and hence does not belong to the proper codomain of σ. $\square$

4.3 Early Congruence

As in the case for bisimilarity, the axiom system for the early congruence is obtained by adding **SP** to the one for the late congruence. Therefore, denoting by $\mathcal{EC}$ the axiom system for early congruence, we have

$$\mathcal{EC} = \mathcal{LC} \cup \{\mathbf{SP}\}$$

Proposition 4.10 (soundness of $\mathcal{EC}$) *If* $\mathcal{EC} \vdash P = Q$ *then* $P \sim_{\mathrm{E}} Q$.

PROOF: Follows from the soundness of **SP** and of the inference rules **IS**-**IC**, the soundness of $\mathcal{LC}$ and the inclusion $\sim_{\mathrm{L}} \subset \sim_{\mathrm{E}}$. $\square$

The completeness proof for $\mathcal{EC}$ is constructed from the one for $\mathcal{LC}$ in the same way as the completeness proof for $\mathcal{EB}$ was constructed from the one for $\mathcal{LB}$, in Section 3.3.

Theorem 4.11 (completeness of $\mathcal{EC}$ for $\sim_{\mathrm{E}}$) *If* $P \sim_{\mathrm{E}} Q$, *then* $\mathcal{EC} \vdash P = Q$.

PROOF: The same schema as in the proof of Theorem 4.9 applies here. Using Lemma 4.8 we can suppose that P and Q are in hnf on $\mathrm{fn}(P, Q)$ and we reason by induction on their depths. The sole difference from Theorem 4.9 is in the inductive step, when considering a summand of P whose first action is an input. We shall only look at this case.

Let $P_{\varphi,a}$ be the sum of all summands $\varphi_i\, \alpha_i.\, P_i$ of P whose outermost condition φ_i is the same as φ modulo semantic equality of conditions and whose outermost prefix α_i is the same as $a(x)$ modulo alpha-conversion and identification of names equated in φ_i; i.e., formally

$$\{\mathbf{C3}, \mathbf{CC1}, \mathbf{CP2}\} \vdash \varphi_i\, \alpha_i.\, P_i = \varphi\, a(x).\, P_i$$

Then we do the same for Q; let

$$P_{\varphi,a} \equiv \textstyle\sum_{i=1}^{n} \varphi\, a(x).P_i \qquad \text{and} \qquad Q_{\varphi,a} \equiv \textstyle\sum_{j=1}^{m} \varphi\, a(x).Q_j$$

be the terms so obtained. We show that $\mathcal{EC} \vdash P_{\varphi,a} = Q_{\varphi,a}$ in the same way as in Theorem 3.5 we proved $\mathcal{EB} \vdash P_a = Q_a$; basically, we shall go through the proof of the latter reformulating those parts which are not true anymore. Therefore the key of the proof is to find, for each $1 \leq i \leq n$, an agent R_i for which it holds that

$$\mathcal{EC} \vdash \varphi\, a(x).\, P_i = \varphi\, a(x).\, R_i \tag{21}$$

$$\mathcal{EC} \vdash Q_{\varphi,a} = Q_{\varphi,a} + \varphi\, a(x).\, R_i \tag{22}$$

Before defining the agent R_i, we have to establish the counterparts of (11). We shall get something slightly weaker, namely (24) and(25) below, but these are enough for our purposes. Let σ be a substitution which agrees with φ. From $P\sigma \mathrel{\dot{\sim}_{\mathrm{E}}} Q\sigma$ we derive $P_{\varphi,a}\,\sigma \mathrel{\dot{\sim}_{\mathrm{E}}} Q_{\varphi,a}\,\sigma$ because, intuitively, $P_{\varphi,a}\,\sigma$ and $Q_{\varphi,a}\,\sigma$ collect exactly the summands of $P\sigma$ and $Q\sigma$ which are able to perform an action labelled $a(x)\sigma$. Therefore, given $P_{\varphi,a}\,\sigma \xrightarrow{a(x)\sigma} P_i\sigma$, by definition of $\dot{\sim}_{\mathrm{E}}$, for each $y \in \mathrm{fn}(P,Q) \cup \{x\}$ there is a $J(i,y)$ s.t. $Q_{\varphi,a}\,\sigma \xrightarrow{a(x)\sigma} Q_{J(i,y)}\sigma$ and

$$P_i\sigma\{y/x\} \mathrel{\dot{\sim}_{\mathrm{E}}} Q_{J(i,y)}\sigma\{y/x\} \tag{23}$$

We would now like to use Lemma 4.5 to lift the above occurrence of $\dot{\sim}_{\mathrm{E}}$ up to $\sim_{\mathrm{E}}$. By definition of hnf, P_i and $Q_{J(i,y)}$ are of the form $\varphi\, P_i'$ and $\varphi\, Q'_{J(i,y)}$, respectively. The condition φ does not mention x and hence may not be complete on $\mathrm{fn}(P_i, Q_{J(i,y)}) = \mathrm{fn}(P,Q) \cup \{x\}$; but we can complete it by adding a conditional at the top which respects $\{y/x\}$. Thus if $V = \mathrm{fn}(P,Q)$ and $[x \notin V] = [\bigwedge_{z\in V}[x \neq z]]$, then (23) and simple algebraic manipulations give

$$\big([(x=y) \wedge \varphi]\, P_i'\big)\sigma\{y/x\} \mathrel{\dot{\sim}_{\mathrm{E}}} \big([(x=y)\wedge\varphi]\, Q'_{J(i,y)}\big)\sigma\{y/x\} \qquad \text{for } y \in V$$

$$\big([(x\notin V) \wedge \varphi]\, P_i'\big)\sigma\{y/x\} \mathrel{\dot{\sim}_{\mathrm{E}}} \big([(x\notin V)\wedge\varphi]\, Q'_{J(i,y)}\big)\sigma\{y/x\} \qquad \text{for } y \notin V$$

Now we apply Lemma 4.5, for $V_1 = V \cup \{x\}$, $\sigma_1 = \sigma\{y/x\}$ and $V_2 = \emptyset$ in the assertion of the lemma (we can assume that the proper domain of $\sigma\{y/x\}$ is contained in $V \cup \{x\}$). Thus we get

$$\big([(x=y) \wedge \varphi]\, P_i'\big) \sim_{\mathrm{E}} \big([(x=y)\wedge\varphi]\, Q'_{J(i,y)}\big) \qquad \text{for } y \in V$$

$$\big([(x\notin V) \wedge \varphi]\, P_i'\big) \sim_{\mathrm{E}} \big([(x\notin V)\wedge\varphi]\, Q'_{J(i,y)}\big) \qquad \text{for } y \notin V$$

from which, using the inductive assumption and **CC1**, we infer

$$\mathcal{EC} \vdash [x=y]\, P_i = [x=y]\, Q_{J(i,y)} \qquad \text{for } y \in \{y_1, \ldots, y_k\} \tag{24}$$

$$\mathcal{EC} \vdash [x\notin V]\, P_i = [x \notin V]\, Q_{J(i,y)} \qquad \text{for } y \notin \{y_1, \ldots, y_k\} \tag{25}$$

Now as in Theorem 3.5, we set $R_i \equiv S_{i,k}$, where for $0 \leq l \leq k$ the agent $S_{i,l}$ is defined in terms of the agents $Q_{J(i,y)}$ as in (12) and (13). Having defined R_i, we have to prove (21) and (22) (the equalities (24) and (25) are used to prove (21)). Now, (21) and (22) are the counterparts of (7) and (8) in Theorem 3.5, respectively. Since the proof of (8) was obtained employing only axioms which are also valid in $\mathcal{EC}$, the same proof can be given for (22), if we first factorise out the conditional φ from the definitions of $Q_{\varphi,a}$ and $\varphi a(x).\, R_i$. Unfortunately the same is not true for for (7), whose derivation in Theorem 3.5 uses **C1** and **C2** (in fact now (14) does not hold anymore). Thus we have to proceed in a different way. For notational simplicity we fix $k = 2$; the generalisation to an arbitrary value for k is immediate. By repeatedly applying **SC2** to decompose the binary conditionals in the definition of $S_{i,k}$ into unary conditionals, and then **SC1**, **CC1** to group conditionals, we get:

$$\mathcal{EC} \vdash R_i =$$
$$[x = y_2]\, Q_{J(i,y_1)} + [(x \neq y_2) \wedge (x = y_1)]\, Q_{J(i,y_2)} + [(x \neq y_2) \wedge (x \neq y_1)]\, Q_{J(i,x)}$$

On the other hand, using **C7**, **C3** and **SC4**, we have

$$\begin{aligned} \mathcal{EC} \vdash P_i &= \mathit{True}\ P_i \\ &= \big[(x = y_2) \vee ((x \neq y_2) \wedge (x = y_1)) \vee ((x \neq y_2) \wedge (x \neq y_1))\big]\, P_i \\ &= [x = y_2]\, P_i + [(x \neq y_2) \wedge (x = y_1)]\, P_i + [(x \neq y_2) \wedge (x \neq y_1)]\, P_i \end{aligned}$$

Now the corresponding summands of the agents obtained from R_i and P_i can be proved equal using (24), (25), and (for the second summand) **CC1**.

Finally, from (21) and (22) we conclude $P_{\varphi,a} = Q_{\varphi,a}$ in the same way as in Theorem 3.5 from (7) and (8) we inferred $P_a = Q_a$. □

5 Deterministic Agents

Definition 5.1 *The* deterministic *agents, in the following ranged over by $P, Q, \ldots$ are here given by*

$$P ::= \mathbf{0} \mid \alpha.\, P \mid \varphi\, P\, Q$$

(In traditional process algebras such as CCS, the so called deterministic agents use "+" in a limited way, disallowing for example $a.\, P + a.\, Q$ while allowing $a.\, P + b.\, Q$ if $a \neq b$. In the present paper we cannot admit "+" at all since the input prefix would then result in nondeterminism — consider $c(a).\, (a.\, P + b.\, Q)$ where the bound a could become identified with the free b.)

IP1	If $P = Q$ then $\overline{a}x.\,P = \overline{a}x.\,Q$	
IP2	If $P\{y/x\} = Q\{y/x\}$ for all $y \in \mathrm{fn}(P,Q,x)$ then $a(x).\,P = a(x).\,Q$	
C1	$\varphi\, P\, Q = P$	if $[\![\varphi]\!] = \mathit{True}$
C2	$\varphi\, P\, Q = Q$	if $[\![\varphi]\!] = \mathit{False}$

Table 3: Axiom system $\mathcal{DB}$ for deterministic agent bisimilarity.

In this section we will only consider deterministic agents, and often write just "agent" for "deterministic agent". The attribute "deterministic" is motivated by the fact that such agents have at most one transition (up to alpha-conversion), and that this property is preserved by transitions:

Proposition 5.2

1. *If* $P \xrightarrow{\alpha} Q$ *and* $P \xrightarrow{\beta} R$ *then* $\alpha.\,Q \equiv \beta.\,R$.

2. *If* $P \xrightarrow{\alpha} Q$ *and* P *is deterministic then* Q *is deterministic.*

Proof: Immediate by induction over agents. □

We then get the following:

Proposition 5.3 $P \mathrel{\dot{\sim}_{\mathrm{L}}} Q$ *iff* $P \mathrel{\dot{\sim}_{\mathrm{E}}} Q$, *and* $P \sim_{\mathrm{L}} Q$ *iff* $P \sim_{\mathrm{E}} Q$.

Proof: In view of Proposition 5.2, an early bisimulation is also a late bisimulation. □

In fact our equivalences also coincide with *trace equivalence*, where two agents are equivalent if they generate the same sequences of communications. For the remainder of this section we will continue to use late bisimilarity as the definition of equivalence, and we will drop the subscripts of $\dot{\sim}$ and $\sim$ to stress the fact the equivalences coincide.

5.1 Bisimilarity

Let $\mathcal{DB}$ be the inference system obtained from $\mathcal{LB}$ by deleting all axioms and rules mentioning $+$. Thus $\mathcal{DB}$ consists of **IP1**, **IP2**, **C1**, and **C2**.

Theorem 5.4 $\mathcal{DB}$ *is sound and complete for bisimilarity on deterministic agents.*

Proof: Each rule is easily proved sound directly from the definition of bisimilarity. For completeness, assume $P \mathrel{\dot{\sim}} Q$, we can prove $\mathcal{DB} \vdash P = Q$ by induction on the sum of the depths of P and Q. The details are straightforward and are omitted. □

IP If $P = Q$ then $\alpha . P = \alpha . Q$

IC$*$ If $P = Q$ then $\varphi \; P \; R \; = \; \varphi \; Q \; R$

C3$*$ if $\varphi \Leftrightarrow \psi$ then $\varphi \; P \; Q \; = \; \psi \; P \; Q$

C4$*$ $\mathit{False} \; Q \; P \; = \; \mathit{False} \; R \; P$

C5 $\varphi \; P \; P \; = P$

C6 $\varphi \; P \; Q \; = \; \neg\varphi \; Q \; P$

CC1$*$ $\varphi \; (\; \psi \; P \; Q \;) \; R \; = \; [\varphi \wedge \psi] \; P \; (\; \varphi \; Q \; R \;)$

CP1 $\varphi \, (\alpha . P) = \; \varphi \, (\alpha . \; \varphi \; P)$ if $\mathrm{bn}(\alpha) \cap \mathrm{n}(\varphi) = \emptyset$

CP2 $[x = y] \; \alpha . P = \; [x = y] \; (\alpha\{x/y\}) . P$

Table 4: Axiom system $\mathcal{DC}$ for deterministic agent congruence.

5.2 Congruence

The axiom system $\mathcal{DC}$ for congruence of deterministic agents is given in Table 4. Basically, $\mathcal{DC}$ is obtained from $\mathcal{LC}$ or $\mathcal{EC}$ by removing all axioms and inference rules mentioning sum and by strengthening some laws to binary conditionals. In particular, **CC1$*$** fulfills two functions: it establishes a relation between a boolean connective ($\wedge$) and conditionals, and it entails a form of associativity of conditionals (apparent by putting $\varphi = \psi$). In contrast, **CC1** only had the first of these functions; the second could be recovered exploiting the associativity of sum. The laws are easily seen to be sound and we concentrate here on completeness. First, some derived laws.

C7 $\mathit{True} \; P \; Q \; = P$

CC1 $\varphi \, (\; \psi \; P) = \; [\varphi \wedge \psi] \; P$

CC2 $\varphi \; P \; (\; \psi \; P) \; = \; [\varphi \vee \psi] \; P$

CC3 $\varphi \; (\; \varphi \; P) \; Q \; = \; \varphi \; P \; Q$

CC4 $\varphi \; P \; (\; \psi \; Q \; R \;) \; = \; \psi \; Q \; (\; \varphi \; P \; R \;)$ if $\psi \wedge \varphi \Leftrightarrow \mathit{False}$

Proof: Note that for **CC4** the side condition implies $\neg\varphi \wedge \psi \Leftrightarrow \psi$.

$$\begin{array}{ll}
\textbf{C7}: & \mathit{True}\; P\; Q \overset{\textbf{C6}}{=} \mathit{False}\; Q\; P \overset{\textbf{C4}*}{=} \mathit{False}\; P\; P \overset{\textbf{C5}}{=} P \\
\textbf{CC1}: & \varphi\,(\,\psi\, P) \overset{\textbf{CC1}*}{=} [\varphi \wedge \psi]\; P\,(\,\varphi\, \mathbf{0}\, \mathbf{0}\,) \overset{\textbf{C5}}{=} [\varphi \wedge \psi]\; P \\
\textbf{CC2}: & \varphi\; P\,(\,\psi\, P) \overset{\textbf{C6}}{=} \neg\varphi\,(\,\neg\psi\; \mathbf{0}\; P\,)\; P \overset{\textbf{CC1}*}{=} \\
 & [\neg\varphi \wedge \neg\psi]\; \mathbf{0}\,(\,\neg\varphi\; P\; P\,) \overset{\textbf{C5}}{=} [\neg\varphi \wedge \neg\psi]\; \mathbf{0}\; P \overset{\textbf{C6},\textbf{C3}*}{=} [\varphi \vee \psi]\; P \\
\textbf{CC3}: & \varphi\,(\,\varphi\, P)\; Q \overset{\textbf{C6}}{=} \varphi\,(\,\neg\varphi\; \mathbf{0}\; P\,)\; Q \overset{\textbf{CC1}*}{=} [\varphi \wedge \neg\varphi]\; \mathbf{0}\,(\,\varphi\; P\; Q\,) \overset{\textbf{C3}*,\textbf{C6}}{=} \\
 & \mathit{True}\,(\,\varphi\; P\; Q\,) \overset{\textbf{C7}}{=} \varphi\; P\; Q \\
\textbf{CC4}: & \varphi\; P\,(\,\psi\; Q\; R\,) \overset{\textbf{C6}}{=} \neg\varphi\,(\,\psi\; Q\; R\,)\; P \overset{\textbf{CC1}*}{=} [\neg\varphi \wedge \psi]\; Q\,(\,\neg\varphi\; R\; P\,) \overset{\textbf{C3}*}{=} \\
 & \psi\; Q\,(\,\neg\varphi\; R\; P\,) \overset{\textbf{C6}}{=} \psi\; Q\,(\,\varphi\; P\; R\,)
\end{array}$$

□

Nested conditionals will play an important part in the completeness proof so we define a derived *case*-construct for convenience:

Definition 5.5

$$[\varphi_1 \Rightarrow P_1, \ldots, \varphi_n \Rightarrow P_n] \;\textit{means}\; \varphi_1\, P_1\,(\ldots(\,\varphi_n\, P_n))$$

Here we also allow $n = 0$ and let the empty case construct [] stand for **0**. We will refer to $\varphi_i \Rightarrow P_i$ as a *branch* of the case construct.

Definition 5.6 *Let V be a set of names. We say that P is in* head normal form *(*hnf*) on V if P is of the form*

$$[\varphi_1 \Rightarrow \alpha_1.\, \varphi_1 P_1, \ldots, \varphi_n \Rightarrow \alpha_n.\, \varphi_n P_n]$$

where for all i, j:

1. $\mathrm{bn}(\alpha_i) \notin V$*;*

2. *φ_i is complete on V;*

3. *If $i \neq j$ then $\varphi_i \wedge \varphi_j \Leftrightarrow$ False.*

The last clause expresses that φ_i and φ_j are *not simultaneously satisfiable*, i.e., no substitution σ can make them both evaluate to *true*.

Lemma 5.7 *For each agent P, and for each finite set of names V with* $\mathrm{fn}(P) \subseteq V$*, there is an agent H, of no greater depth than P, in hnf on V s.t. $\mathcal{DC} \vdash P = H$.*

Proof: By structural induction over agents. **0** is already in hnf. The inductive step has two cases.

The first case is that $P \equiv \alpha. P'$. First, use **A** to insure that a name bound by α (if any) is not in V. Let $\mathcal{R}_1, \ldots, \mathcal{R}_n$ enumerate all possible equivalence relations on V, and let φ_i be the complete condition corresponding to $\mathcal{R}_i$. Thus,

$$\bigvee_{i=1}^{n} \varphi_i \Leftrightarrow \mathit{True} \tag{26}$$

and moreover for all $i \neq j$, $\varphi_i \wedge \varphi_j \Leftrightarrow \mathit{False}$. Define

$$H \equiv [\varphi_1 \Rightarrow \alpha. \varphi_1 P', \ldots, \varphi_n \Rightarrow \alpha. \varphi_n P']$$

Clearly H is in hnf. We will prove $\mathcal{DC} \vdash P = H$. First use **C7** to get $P = \mathit{True}\ P$, then use (26) and **C3**$*$ to rewrite this to $[\bigvee_{i=1}^{n} \varphi_i]\, P$; thereafter n applications of **CC2**, **CC3** and **CP1** gives

$$[\varphi_1 \Rightarrow \ \varphi_1\, \alpha.\ \varphi_1\ P', \ldots, \varphi_n \Rightarrow \ \varphi_n\ \alpha.\ \varphi_n P']$$

and finally using **CC3** in the other direction on each of the branches we obtain H.

The second case is that P is a conditional, $P \equiv \ \varphi\, Q\ R$. By induction Q and R can be written on hnf on V:

$$\mathcal{DC} \vdash Q = [\varphi_1 \Rightarrow Q_1, \ldots, \varphi_n \Rightarrow Q_n]\ ; \quad R = [\psi_1 \Rightarrow R_1, \ldots, \psi_m \Rightarrow R_m]$$

We now claim that P is provably equivalent with a hnf

$$[\chi_1 \Rightarrow S_1, \ldots, \chi_p \Rightarrow S_p]$$

where the χ_i additionally satisfy the following requirement which we call (†): For each i either $\chi_i \Leftrightarrow \varphi_j \Leftrightarrow \varphi \wedge \varphi_j$ for some j, or $\chi_i \Leftrightarrow \psi_k \Leftrightarrow \neg\varphi \wedge \psi_k$ for some k, and further each φ_j and ψ_k correspond to at most one χ_i in this way. We will prove the claim by induction on $n + m$.

The base case is that $n = m = 0$, then $Q \equiv R \equiv \mathbf{0}$, so $P \overset{\mathbf{C5}}{=} \mathbf{0}$. For the inductive step $n > 0$ or $m > 0$; assume first $n > 0$. Then

$$P \equiv \ \varphi\ (\varphi_1 Q_1 [\varphi_2 \Rightarrow Q_2, \ldots, \varphi_n \Rightarrow Q_n])\ R$$

Applying **CC1**$*$ gives

$$P \overset{\mathbf{CC1}*}{=} [\varphi \wedge \varphi_1]\ Q_1\ (\ \varphi\ [\varphi_2 \Rightarrow Q_2, \ldots, \varphi_n \Rightarrow Q_n]\ R\ )$$

By the inner induction (on $n+m$) the expression within parentheses is provably equivalent with a hnf $[\chi_1 \Rightarrow S_1, \ldots, \chi_q \Rightarrow S_q]$, satisfying (†). Thus we get

$$\mathcal{DC} \vdash P = [\varphi \wedge \varphi_1]\, Q_1\, [\chi_1 \Rightarrow S_1, \ldots, \chi_q \Rightarrow S_q]$$

Since φ_1 is complete on V there are now, by Lemma 4.3.2, two possibilities. Either $\varphi \wedge \varphi_1$ is unsatisfiable, i.e. $\varphi \wedge \varphi_1 \Leftrightarrow \mathit{False}$. Then **C6** and **C7** reduce P to

$$[\chi_1 \Rightarrow S_1, \ldots, \chi_q \Rightarrow S_q]$$

which is the desired hnf. The other possibility is that $\varphi \wedge \varphi_1 \Leftrightarrow \varphi_1$. Then by **C3**∗

$$\mathcal{DC} \vdash P = [\varphi_1 \Rightarrow Q_1,\ \chi_1 \Rightarrow S_1, \ldots, \chi_q \Rightarrow S_q]$$

where the right hand side is the desired hnf. Clearly, this agent satisfies clauses 1 and 2 in the definition of hnf since Q and R are hnf's. Moreover the agent satisfies (†) since the first condition φ_1 is equivalent to $\varphi \wedge \varphi_1$ and the other conditions satisfy (†) by induction. Clause 3 for hnf follows from (†) since any conjunction of different conditions in the agent is either $\varphi_i \wedge \varphi_j$ for $i \neq j$ which is equivalent to *False* since Q is a hnf, or it is $\varphi_i \wedge \psi_j$ and then

$$\varphi_i \wedge \psi_j \Leftrightarrow (\varphi \wedge \varphi_i) \wedge (\neg\varphi \wedge \psi_j) \Leftrightarrow \mathit{False}.$$

Finally the case $m > 0$ is symmetric to the case $n > 0$ by first applying **C6** to obtain $P = \neg\varphi\ R\ Q$ and then proceeding as above with $\neg\varphi$ for φ and the roles of Q and R exchanged. □

Theorem 5.8 *If $P \sim Q$ then $\mathcal{DC} \vdash P = Q$.*

PROOF: The proof is by induction on the depth of P and Q. By Lemma 5.7 it is enough to prove the theorem when P and Q are in hnf on $V = \mathrm{fn}(P, Q)$, and by using **A** we can assume that all bound names in all top level input prefixes are the same, say x. For the base case $P \equiv Q \equiv \mathbf{0}$ and we are done. For the inductive step let $\varphi \Rightarrow \alpha.\, P'$ be a branch of P. We shall first show that there exists a branch $\psi \Rightarrow \beta.\, Q'$ of Q s.t. $\mathcal{DC} \vdash \varphi\, \alpha.\, P' = \psi\, \beta.\, Q'$ and $\varphi \Leftrightarrow \psi$.

Since P is in hnf, and hence φ is complete on $\mathrm{fn}(P, Q)$, φ is satisfiable; so there exists a substitution σ which agrees with φ. It follows that $P\sigma \xrightarrow{\alpha\sigma} P'\sigma$. Since $P \sim Q$ there must be a similar transition from $Q\sigma$. Evidently this can only be generated by a branch $\psi \Rightarrow \beta.\, Q'$ where ψ also agrees with σ, and since Q is in hnf there can only be one such branch in Q. From this, Lemma 4.3.3 and the definition of $\dot\sim$ we have:

(a) $\psi \Leftrightarrow \varphi$,

(b) $\beta\sigma = \alpha\sigma$,

(c) if α, β are outputs, then $P'\sigma \mathrel{\dot{\sim}} Q'\sigma$,

(d) if α, β are inputs (with bound name x), then for all z we have $P'\sigma\{^{z}\!/\!_{x}\} \mathrel{\dot{\sim}} Q'\sigma\{^{z}\!/\!_{x}\}$.

We now reason as in the proof of Theorem 4.9 and exploit **CP2** to infer $\mathcal{DC} \vdash \varphi\,\beta.\,Q' = \varphi\,\alpha.\,Q'$, and Lemma 4.5 to establish that $\mathcal{DC} \vdash \varphi\,\alpha.\,P' = \psi\,\beta.\,Q'$. (This part of the proof of Theorem 4.9 does not use summation and carries over to $\mathcal{DC}$). Symmetrically, for each branch in Q there is a corresponding branch in P satisfying this and $\varphi \Leftrightarrow \psi$. The conditions in the branches are not simultaneously satisfiable and hence with **CC4** we can reorder the branches in P and Q so that corresponding branches occur in the same relative positions; then with **C3**$*$ we can make the conditions identical. That is, if

$$P \equiv [\varphi_1 \Rightarrow P_1, \ldots, \varphi_n \Rightarrow P_n]$$

we get

$$\mathcal{DC} \vdash Q = [\varphi_1 \Rightarrow Q_1, \ldots, \varphi_n \Rightarrow Q_n]$$

where $\mathcal{DC} \vdash \varphi_i\,P_i = \varphi_i\,Q_i$ for each i. By repeatedly applying the following to each branch of P:

$$\varphi_i\,P_i\,R \overset{\mathbf{CC3}}{=} \varphi_i\,(\,\varphi_i\,P_i)\,R \overset{\mathcal{DC}}{=} \varphi_i\,(\,\varphi_i\,Q_i)\,R \overset{\mathbf{CC3}}{=} \varphi_i\,Q_i\,R$$

we can transform P into Q. This concludes the inductive step and the completeness proof. □

6 Independence

The axiom systems in this paper are far from trivial, and it is natural to ask if the complexity is necessary or if there are simpler axiomatisations. Partial answers to this question can be given by establishing the independence of individual axioms. Given an axiom system $\mathcal{AS}$, we say that an axiom in $\mathcal{AS}$ is *independent* if some instance of it cannot be inferred from the rest of $\mathcal{AS}$. Thus, the removal of an independent axiom affects provable equality in $\mathcal{AS}$.

We show in this section that each of our axiom systems consists of independent axioms. We only explicitly consider the system $\mathcal{LC}$ for late congruence. The independence of the axioms in $\mathcal{LB}$ is evident. Moreover the argument used for the late systems carries over to the early and deterministic ones. The independence of the additional axiom

SP in the early systems is proved by the difference between the late and the early bisimilarities and congruences.

The major problems of establishing independence in $\mathcal{LC}$ are caused by the two corresponding groups of axioms for sum and conditionals, namely **S1**-**S4** and **C4**, **C5**, **C6**, **CC1**; as pointed out in Section 4.2 these axioms express similar properties, like idempotency and commutativity, although for different operators. Thus some (but not all) instances of these axioms are interderivable in subtle ways. For instance, consider the associative law **S4** for sum; in $\mathcal{LC} - \mathbf{S4}$ we can prove:

$$
\begin{array}{l}
(\,\varphi\, P + \neg\varphi\, Q) + \neg\varphi\, R \;\overset{\mathbf{S1},\mathbf{S3},\mathbf{C5}}{=}\; (\,\varphi\, P + \neg\varphi\, Q) + (\,\varphi\, \mathbf{0} + \neg\varphi\, R) \;\overset{\mathbf{SC2}}{=} \\
\varphi\, P\, Q \,+\, \varphi\, \mathbf{0}\, R \;\overset{\mathbf{SC1*},\mathbf{S1}}{=}\; \varphi\, P\, (Q + R) \;\overset{\mathbf{SC2}}{=} \\
\varphi\, P + \neg\varphi\, (Q + R) \;\overset{\mathbf{SC1}}{=}\; \varphi\, P + (\,\neg\varphi\, Q + \neg\varphi\, R)
\end{array}
$$

This is an instance of **S4**.

We here first examine the group of axioms **S1**-**S4**, then the corresponding axioms for conditionals **C4**, **C5**, **C6**, **CC1**, and finally the remaining axioms **C3**, **SC1***, **CP1**, **CP2**. We do not explicitly consider the inference laws (these are obviously independent).

Lemma 6.1 *Each of the axioms* **S1**-**S4** *is independent in* $\mathcal{LC}$.

PROOF: We exploit a reduction operation $P^{\succ}$ on agents (we assign the symbol $\succ$ precedence over the operators of the language). Intuitively, $P^{\succ} = P'$ means that P becomes P' when evaluating all conditions in P. We require however that P is *only-emitting*, that is for each substitution σ, the agent $P\sigma$ and all its derivatives can only perform output actions. The motivation for this restriction is that it does not make sense to evaluate a condition underneath an input prefix, since some of the names in the condition might be bound by the input. Note that an agent might be only-emitting and yet have input prefixes in its syntactic form, like *False* $a(x).\,\mathbf{0}$. Furthermore, by the soundness of $\mathcal{LC}$, if $\mathcal{LC} \vdash P = Q$, and P is only-emitting, then also Q is only-emitting. Formally, $\succ$ is defined on only-emitting agents as follows ($[\![\,]\!]$ is the evaluation of conditions presented in Section 2):

$$
\begin{array}{rcl}
(\,\varphi\, P\, Q\,)^{\succ} & = & \begin{cases} P^{\succ} & \text{if } [\![\varphi]\!] = \mathit{True} \\ Q^{\succ} & \text{if } [\![\varphi]\!] = \mathit{False} \end{cases} \\
(P + Q)^{\succ} & = & P^{\succ} + Q^{\succ} \\
\mathbf{0}^{\succ} & = & \mathbf{0} \\
(\alpha.\, P)^{\succ} & = & \alpha.\, P^{\succ}
\end{array}
$$

The reduction $\succ$ will be used to eliminate from a derivation in $\mathcal{LC}$ the effects of the axioms involving conditionals. For each axiom **Si**, we give a property on $\succ$ which is invariant in the system $\mathcal{LC}$ - **Si**, but which fails for **Si**. Every invariance can be proved by an easy induction on the length of a derivation. Just consider that if two only-emitting agents P, Q are equated by any rule different from **S1** – **S4**, then $P^{\succ} = Q^{\succ}$: This fact holds for **IP** because the agents are only-emitting and hence α cannot be an input; all remaining rules require a straightforward case analysis (for instance, take **CC1**: If $[\![\varphi \wedge \psi]\!] = \mathit{True}$, then we have $(\,\varphi\,(\,\psi\,P\,)\,)^{\succ} = (\,[\varphi \wedge \psi]\;P\,)^{\succ} = P^{\succ}$, otherwise, if $[\![\varphi \wedge \psi]\!] = \mathit{False}$, then $(\,\varphi\,(\,\psi\,P\,)\,)^{\succ} = (\,[\varphi \wedge \psi]\;P\,)^{\succ} = \mathbf{0}$).

We consider each axiom in order.

S1: Let us call a subterm unguarded if is not underneath a prefix. We have:

if $\mathcal{LC} - \mathbf{S1} \vdash P = Q$ and $P^{\succ}$ has an unguarded $\mathbf{0}$ subterm,
then so has $Q^{\succ}$.

It follows that the instance $\alpha + \mathbf{0} = \alpha$ of **S1** is not derivable in $\mathcal{LC} - \mathbf{S1}$, for $\alpha + \mathbf{0}$ and α reduce under $\succ$ to themselves, but the only former has an unguarded occurrence of $\mathbf{0}$.

S2: We say that a term $\sum_i \alpha_i.\,P_i$ has multiplicity, if there are $i_1 \neq i_2$, with $\alpha_{i_1} = \alpha_{i_2}$. Then it holds that:

if $\mathcal{LC} - \mathbf{S2} \vdash P = Q$ and $P^{\succ}$ has multiplicity, then so has $Q^{\succ}$.

The agents $\alpha + \alpha$ and α are equated by **S2**. They reduce under $\succ$ to themselves, but only the former has multiplicity. Hence they are not equated in $\mathcal{LC} - \mathbf{S2}$.

S3: Consider the following predicate $\mathtt{B_{S3}}(P)$, defined on conditional-free agents; $\mathtt{B_{S3}}(P)$ holds if in P there is a path leading to a prefix and, in a sum, the path always selects the left arm:

$$\begin{aligned} \mathtt{B_{S3}}(\mathbf{0}) &= \mathit{False} \\ \mathtt{B_{S3}}(\alpha.\,P) &= \mathit{True} \\ \mathtt{B_{S3}}(P + Q) &= \mathtt{B_{S3}}(P) \end{aligned}$$

It holds that

if $\mathcal{LC} - \mathbf{S3} \vdash P = Q$, then $\mathtt{B_{S3}}(P^{\succ}) = \mathtt{B_{S3}}(Q^{\succ})$

As a consequence, the instance $\alpha + \mathbf{0} = \mathbf{0} + \alpha$ of **S3** is not derivable in $\mathcal{LC} - \mathbf{S3}$, for $\alpha + \mathbf{0}$ and $\mathbf{0} + \alpha$ reduce under $\succ$ to themselves, but $\mathtt{B_{S3}}(\alpha + \mathbf{0})$ holds, whereas $\mathtt{B_{S3}}(\mathbf{0} + \alpha)$ does not.

S4: We say that an agent P is $\{\alpha, \beta\}$-active if $P \xrightarrow{\alpha}$ and $P \xrightarrow{\beta}$ hold, but $P \xrightarrow{\gamma}$ does not, for any $\gamma \neq \alpha, \beta$. We have

> if $\mathcal{LC} - \mathbf{S3} \vdash P = Q$ and $P^{\succ}$ has an $\{\alpha, \beta\}$-active subterm,
> then so has $Q^{\succ}$.

Now, take the instance $\gamma + (\alpha + \beta) = (\gamma + \alpha) + \beta$ of **S4**, for $\gamma \neq \alpha, \beta$. The agents $\gamma + (\alpha + \beta)$ and $(\gamma + \alpha) + \beta$ reduce under $\succ$ to themselves, but only the former has an $\{\alpha, \beta\}$-active subterm. We conclude that **S4** is independent in $\mathcal{LC}$.

□

Lemma 6.2 *The axioms* **C4**, **C5**,**C6** *and* **CC1** *are independent in* $\mathcal{LC}$ *.*

PROOF: **C4** and **C5** are independent because they are the only axioms which allow us to modify the depth of an agent and to introduce a conditional. For **CC1** we consider an agent transformation $\mathsf{R}_{\mathsf{CC1}}(P)$ which, intuitively, replaces every true conditional with its left arm and every false conditional with its right arm. Formally

$$
\begin{aligned}
\mathsf{R}_{\mathsf{CC1}}(\mathbf{0}) &= \mathbf{0} \\
\mathsf{R}_{\mathsf{CC1}}(\alpha.\,P) &= \alpha.\,\mathsf{R}_{\mathsf{CC1}}(P) \\
\mathsf{R}_{\mathsf{CC1}}(P + Q) &= \mathsf{R}_{\mathsf{CC1}}(P) + \mathsf{R}_{\mathsf{CC1}}(Q) \\
\mathsf{R}_{\mathsf{CC1}}(\ \varphi\ P\ Q\ ) &= \begin{cases} \mathsf{R}_{\mathsf{CC1}}(P) & \text{if } \varphi \Leftrightarrow \mathit{True} \\ \mathsf{R}_{\mathsf{CC1}}(Q) & \text{if } \varphi \Leftrightarrow \mathit{False} \\ \ \varphi\ \mathsf{R}_{\mathsf{CC1}}(P)\ \mathsf{R}_{\mathsf{CC1}}(Q) & \text{otherwise} \end{cases}
\end{aligned}
$$

It holds that

> if $\mathcal{LC} - \mathbf{CC1} \vdash P = Q$ and in $\mathsf{R}_{\mathsf{CC1}}(P)$ there is a prefix,
> then also in $\mathsf{R}_{\mathsf{CC1}}(Q)$ there is a prefix.

This can be proved by induction on the length of the inference of $P = Q$. Therefore, the instance $\ \varphi\ (\ \neg\varphi\ \alpha) = \ [\varphi \wedge \neg\varphi]\ \alpha$ is not derivable in $\mathcal{LC} - \mathbf{CC1}$, since $\mathsf{R}_{\mathsf{CC1}}(\ \varphi\ (\ \neg\varphi\ \alpha)) = \ \varphi\ (\ \neg\varphi\ \alpha)$ has a prefix, but $\mathsf{R}_{\mathsf{CC1}}(\ [\varphi \wedge \neg\varphi]\ \alpha) = \mathbf{0}$ has not.

The independence proof of **C6**, finally, is similar to the proof for **S3** in the previous lemma. Consider the predicate $\mathsf{B}_{\mathsf{C6}}(P)$ inductively defined below. Intuitively, $\mathsf{B}_{\mathsf{C6}}(P)$ holds if in P there is a path leading to a prefix and, in a conditional, the path always selects the right arm.

$$
\begin{aligned}
\mathsf{B}_{\mathsf{C6}}(\mathbf{0}) &= \mathit{False} \\
\mathsf{B}_{\mathsf{C6}}(\alpha.\,P) &= \mathit{True} \\
\mathsf{B}_{\mathsf{C6}}(P + Q) &= \mathsf{B}_{\mathsf{C6}}(P) \vee \mathsf{B}_{\mathsf{C6}}(Q) \\
\mathsf{B}_{\mathsf{C6}}(\ \varphi\ P\ Q\ ) &= \mathsf{B}_{\mathsf{C6}}(Q)
\end{aligned}
$$

It holds that

$$\mathcal{LC} - \mathbf{C6} \vdash P = Q \quad \text{implies } \mathsf{B}_{\mathsf{C6}}(P) = \mathsf{B}_{\mathsf{C6}}(Q)$$

It follows that the instance $\varphi\ \alpha\ \mathbf{0} = \neg\varphi\ \mathbf{0}\ \alpha$ of **C6** is not derivable in $\mathcal{LC} - \mathbf{C6}$, for $\mathtt{B}_{\mathtt{C6}}(\ \neg\varphi\ \mathbf{0}\ \alpha\ )$ holds, whereas $\mathtt{B}_{\mathtt{C6}}(\ \varphi\ \alpha\ \mathbf{0}\ )$ does not. □

Lemma 6.3 *The axioms* **C3**, **SC1∗**, **CP1**, *and* **CP2** *are independent in* $\mathcal{LC}$ *.*

Proof: The proof is straightforward for **C3** and **CP2**, since they are the only axioms which allow us to modify the syntax of a condition and of a prefix.

Consider **SC1∗**. This axiom expresses the relationship between conditionals and sums. We shall prove that without **SC1∗** a binary conditional cannot be decomposed into the sum of unary conditionals. Let $\mathtt{B}_{\mathtt{SC1*}}(P)$ be a predicate defined inductively on the structure of P as follows:

$$\begin{array}{rcl}
\mathtt{B}_{\mathtt{SC1*}}(\mathbf{0}) & = & \mathit{False} \\
\mathtt{B}_{\mathtt{SC1*}}(\alpha.\,P) & = & \mathit{False} \\
\mathtt{B}_{\mathtt{SC1*}}(P+Q) & = & \mathtt{B}_{\mathtt{SC1*}}(P) \vee \mathtt{B}_{\mathtt{SC1*}}(Q) \\
\mathtt{B}_{\mathtt{SC1*}}(\ \varphi\ P\ Q\ ) & = & \begin{cases} \mathit{True} & \text{if } \varphi\ P\ Q \not\sim \mathbf{0} \text{ and} \\ & \quad \big(\ \mathtt{B}_{\mathtt{SC1*}}(P) \text{ or } \mathtt{B}_{\mathtt{SC1*}}(Q) \text{ or } P \sim \mathbf{0} \text{ or } Q \sim \mathbf{0}\ \big) \\ \mathit{False} & \text{otherwise} \end{cases}
\end{array}$$

Intuitively, $\mathtt{B}_{\mathtt{SC1*}}(P)$ holds if going down into the structure of P there is a conditional which is active (i.e. it can cause actions), and semantically unary (i.e. one of its arms is inactive); however, the search does not continue inside an inactive conditional. We claim that

$$\mathcal{LC} - \mathbf{SC1*} \vdash P = Q \quad \text{implies } \mathtt{B}_{\mathtt{SC1*}}(P) = \mathtt{B}_{\mathtt{SC1*}}(Q)$$

As usual, the proof of the claim proceeds by induction on the length of the inference for $P = Q$. The basis is trivial. For the inductive part, we explicitly examine the cases in which the last rule used is **IC**, **C5** or **CC1**. For **IC**, suppose that $P = Q$ is derived with a shorter inference, and that $\mathtt{B}_{\mathtt{SC1*}}(\ \varphi\, P)$ holds. Therefore it must hold that $\varphi\, P \not\sim \mathbf{0}$; by the soundness of the laws, then also $\varphi\, Q \not\sim \mathbf{0}$, from which, by definition, we get $\mathtt{B}_{\mathtt{SC1*}}(\ \varphi\, Q) = \mathit{True}$. For **C5**, suppose that $\mathtt{B}_{\mathtt{SC1*}}(P)$ holds. Therefore $P \not\sim \mathbf{0}$, for, otherwise, $\mathtt{B}_{\mathtt{SC1*}}(P)$ would be false. But then also $\varphi\ P\ P \not\sim \mathbf{0}$, which together with $\mathtt{B}_{\mathtt{SC1*}}(P) = \mathit{True}$, gives $\mathtt{B}_{\mathtt{SC1*}}(\ \varphi\ P\ P\ ) = \mathit{True}$. For the other direction suppose $\mathtt{B}_{\mathtt{SC1*}}(\ \varphi\ P\ P\ )$ holds. By definition, this means that $\varphi\ P\ P \not\sim \mathbf{0}$ and that $\mathtt{B}_{\mathtt{SC1*}}(P) = \mathit{True}$ or $P \sim \mathbf{0}$. It cannot be the latter, for, otherwise, $\varphi\ P\ P \sim \mathbf{0}$. Hence $\mathtt{B}_{\mathtt{SC1*}}(P)$ must be true. Finally, for **CC1**, just note that $\varphi\ (\ \psi\ P)$ and $[\varphi \wedge \psi]\ P$ are congruent and both have a **0**-arm. Therefore, if one of them is congruent to **0**, then $\mathtt{B}_{\mathtt{SC1*}}$ is false for both; otherwise $\mathtt{B}_{\mathtt{SC1*}}$ is true for both.

Having proved the above claim, we can conclude the independence of **SC1**$*$ as follows: $\varphi\,(\alpha + \mathbf{0})\;(\mathbf{0} + \alpha) \;=\; \varphi\;\alpha\;\mathbf{0} \,+\, \varphi\;\mathbf{0}\;\alpha$ is an instance of **SC1**$*$, but it is not derivable in $\mathcal{LC} - \mathbf{SC1}*$, because the predicate $\mathtt{B}_{\mathtt{SC1*}}$ holds on only on the right hand side.

There remains the independence of **CP1**. Intuitively, this is the only axiom which can affect the congruence of agents underneath an active prefix. Take the predicate $\mathtt{B}_{\mathtt{CP1}}(P)$, which is true if there are P' and α with $a, b \notin \mathrm{bn}(\alpha)$ s.t. $P \xrightarrow{\alpha} P'$ and $P'\{a/b\} \sim \mathbf{0}$. It holds that

$$\mathcal{LC} - \mathbf{CP1} \vdash P = Q \quad \text{implies } \mathtt{B}_{\mathtt{CP1}}(P) = \mathtt{B}_{\mathtt{CP1}}(Q)$$

The proof is by induction on the length of the inference. All cases are straightforward. However, the property fails for **CP1**: for instance, $\mathtt{B}_{\mathtt{CP1}}(\,[a\neq b]\,\alpha.\,\alpha)$ is false and $\mathtt{B}_{\mathtt{CP1}}(\,[a\neq b]\,\alpha.\;[a\neq b]\;\alpha)$ is true, but $[a\neq b]\,\alpha.\,\alpha \;=\; [a\neq b]\;\alpha.\;[a\neq b]\;\alpha$ is an instance of **CP1**. □

In a similar way we can also show that instances of **CP1** with matchings (rather than mismatchings) are independent.

Remark 6.4 The independence argument for **C5** breaks down if **C4** is replaced by:

$$\mathit{False}\;Q = \mathbf{0}$$

since then **C5** would no longer be the only axiom that introduces a conditional. However, **C5** does remain independent. The intuition of the proof is that **C5** would be the only law which can introduce a conditional $\varphi\;P\;Q$ where both P and Q are active (that is, there are substitutions σ_1 and σ_2 s.t. $(\,\varphi\;P\;Q\,)\sigma_1$ uses the subterm P to perform an action, whereas $(\,\varphi\;P\;Q\,)\sigma_2$ uses the subterm Q). With a similar argument, it can be shown that in the system $\mathcal{LC}$, **C5** cannot be replaced by the simpler $\varphi\;\mathbf{0}\;\mathbf{0}\;=\mathbf{0}$. □

Independence issues have hitherto not been much considered in process algebra, in spite of the sometimes complex axiomatisations of some varieties. Partly this may be because independence problems are genuinely difficult. A standard approach to independence in general is to investigate counter models (i.e. objects which are models of all axioms and inference rules except the axiom to be proved independent). For example, **SP** is independent since it is unsound for the late equivalences. In contrast, our proofs in the three lemmas above were arrived at by investigating a large number of derivations and determining properties preserved by inference; we have not found any simpler argument in terms of counter models. A similar type of argument can be found in Faron Moller's PhD Thesis [Mol89] for demonstrating nonexistence of finite axiomatisations.

Although we have proved that the existence of each of our axioms is necessary, some questions might remain on whether some axioms can be weakened. The most interesting question is the replacement in $\mathcal{DC}$ of **CC1**$*$ with the simpler

$$[\varphi \wedge \psi]\, P\, Q \;=\; \varphi\,(\,\psi\, P\, Q\,)\, Q$$

We have not been able to derive **CC1**$*$ from it, nor to prove **CC1**$*$ independent in the presence of it.

7 Extending the Signature

7.1 Restriction

The operator perhaps most responsible for the expressive power of the π-calculus is *restriction.* Restriction of the name x in the agent P is written $(x)\,P$ (in some works it is written $\boldsymbol{\nu} x P$). It is used to make the name x in P inaccessible to the environment of P. However in π-calculus — unlike in CCS — restrictions may disappear; the environment may get acquaintance of x through name-passing.

Restriction is a binding operator and as such subject to alpha-conversion. As a syntactic form, we assign to restriction the same precedence as prefixing. We abbreviate the expression $(x)\overline{a}x.\,P$ to $\overline{a}(x).\,P$ if $a \neq x$. This prefix is called *bound output* and represents the output of a restricted name. Bound outputs are needed in the definitions of the transitional semantics and bisimulations.

To extend the transition relations of Definition 2.2, α has to range also over bound outputs[1] and two rules for restriction must be added:

$$\frac{P \xrightarrow{\alpha} P'}{(x)\,P \xrightarrow{\alpha} (x)\,P'}\; x \notin n(\alpha) \qquad\qquad \frac{P \xrightarrow{\overline{a}x} P'}{(x)\,P \xrightarrow{\overline{a}(x)} P'}\; x \neq a$$

The definitions of bisimulations (Definitions 2.3 and 2.4) are extended with the following clause for bound outputs :

3. If $P \xrightarrow{\overline{a}(x)} P'$ *and* $x \notin \mathrm{fn}(P,Q)$*, then for some* Q'*,* $Q \xrightarrow{\overline{a}(x)} Q'$ *and* $P'\mathcal{S}Q'$*.*

We propose the axioms in Table 5 for restriction. **IR** is the obvious inference law. **R** – **RC2** describe the relationship between restriction and the other operators. Note in particular how restriction interacts with the matching, in axioms **RC1** and **RC2**. If x is restricted then no substitution can make it equal to another name y, hence in **RC1** the evaluation of $[x = y]$ would always return *False*. Conversely, in **RC2** the restriction can be pushed inside the matching since its names are not bound by x.

---

[1] this is not strictly necessary in the rule for prefix.

IR	if $P = Q$ then $(x)\,P = (x)\,Q$	
R	$(x)\,\mathbf{0} = \mathbf{0}$	
RR	$(x)\,(y)\,P = (y)\,(x)\,P$	
RS	$(x)\,(P + Q) = (x)\,P + (x)\,Q$	
RP1	$(x)\,\alpha.\,P = \alpha.\,(x)\,P$	if $x \notin \mathrm{n}(\alpha)$
RP2	$(x)\,\alpha.\,P = \mathbf{0}$	if x is the port of α
RC1	$(x)\,[x=y]\,P = \mathbf{0}$	if $x \neq y$
RC2	$(x)\,[z=y]\,P = [z=y]\,(x)P$	if $x \neq y, z$

Table 5: The axioms for restriction.

The axioms are valid in all four equivalences studied in this paper. All but **RC1** and **RC2** have been presented in [MPW92]. These two axioms are not necessary for completeness of the axiomatisation of bisimilarity, in view of **C1**, **C2**, **IR** and **R**. On the other hand, in the axiomatisations of the congruences **R** is redundant:

$$(x)\,\mathbf{0} \overset{\mathbf{C5}}{=} (x)\,[x=y]\,\mathbf{0} \overset{\mathbf{RC1}}{=} \mathbf{0}$$

The counterparts of **RC1** and **RC2** for mismatching, namely

RC3	$(x)\,[x \neq y]\,P = (x)P$	if $x \neq y$
RC4	$(x)\,[y \neq z]\,P = [y \neq z]\,(x)P$	if $x \neq y, z$

are easily derivable from the axioms for the bisimilarities using **C1**, **C2**, **IR** and **R**. Let us see their derivation in the systems for the congruences. For **RC3** we have:

$$(x)P \overset{\mathbf{SC3,RS}}{=} (x)\,[x=y]\,P + (x)\,[x \neq y]\,P \overset{\mathbf{RC1,S3,S1}}{=} (x)\,[x \neq y]\,P$$

For **RC4** we use the law

$$\mathbf{CC5} \qquad [x=y]\,(\,[x \neq y]\,P) = \mathbf{0}$$

obtained applying **CC1** and **C3** to rewrite the agent on the left as *False P* and then **C4**, **C5** to reduce it to **0**. The agent $(x)\,[y \neq z]\,P$ in **RC4** can be rewritten as follows:

$$\begin{aligned}
&(x)\,[y \neq z]\,P \overset{\mathbf{SC3}}{=} [y=z]\,(x)\,[y \neq z]\,P + [y \neq z]\,(x)\,[y \neq z]\,P \overset{\mathbf{RC2}}{=} \\
&(x)\,[y=z]\,(\,[y \neq z]\,P) + [y \neq z]\,(x)\,[y \neq z]\,P \overset{\mathbf{CC5,R}}{=} \\
&\mathbf{0} + [y \neq z]\,(x)\,[y \neq z]\,P \overset{\mathbf{S3,S1}}{=} [y \neq z]\,(x)\,[y \neq z]\,P
\end{aligned}$$

The other agent in **RC4**, namely $[y \neq z]\,(x)P$, can similarly be rewritten to the agent $[y \neq z]\,(x)\,[y \neq z]\,P$ by expanding P with **SC3**.

By expressing a conditional as sum of elementary conditionals and using **RC1** – **RC4**, we can derive all concrete instances of the law below, which expresses in a single line the relationship between restriction and conditionals. The function Remove_x when applied to the condition φ, replaces each matching $[x = y]$ or $[y = x]$, for $x \neq y$, with *False* and each matching $[x = x]$ with *True*. The resulting condition $\mathit{Remove}_x(\varphi)$ does not contain any occurrence of x and can hence be propagated through the scope of x. The law is:

$$\textbf{RC5} \quad (x)(\,\varphi\, P\ Q\,) = \ \mathit{Remove}_x(\varphi)\ (x)\,P\ \ (x)\,Q$$

In the completeness proofs, the axioms for restriction are only employed in the derivation of hnf's, to push a restriction inside a term (**RR** – **RP1** and **RC5**) until either it disappears (**R** or **RP2**) or it gives rise to a bound output. The term so obtained can be described by the grammar of Definition 2.1, if α is taken to range also over bound outputs. As a consequence, the completeness theorems that we have seen remain virtually unchanged. (When comparing equivalent hnf's in the proofs of Theorems 4.11 and 4.9, bound outputs can be dealt with in the same way as free outputs, provided that alpha-conversion is used to to impose that the outermost bound names are identical.) The only meaningful adjustment is in the definition of hnf for the congruences (Definition 4.7). Remember that the definition was parametrised over a set V of names. The new format is

$$\sum_{i \in I} \varphi_i\ \alpha_i.\ \varphi_i'\ P_i$$

where as before

1. $\mathrm{bn}(\alpha_i) \notin V$,

2. φ_i is complete on V,

3. $\varphi_i = \varphi_i'$ if α is an input or a free output;

but now, if α_i is a bound output, say $\alpha_i = \overline{a}(x)$, then we require that

4. $\varphi_i' = \varphi_i \wedge (\bigwedge_{z \in V}(x \neq z))$

The new clause expresses a consistency requirement. The restriction declares x as a new name never to be confused with any other known name. This essential information must be preserved together with φ_i. To see how to add the mismatchings to φ_i, take the term $\overline{a}(x).\,Q$; we can generate the condition $[x \neq z]$ at the top of Q as follows:

$$(x)\overline{a}x.\,Q \ \overset{\textbf{RC3}}{=}\ (x)\,[x \neq z]\,\overline{a}x.\,Q \ \overset{\textbf{CP1}}{=}\ (x)\,[x \neq z]\,\overline{a}x.\ [x \neq z]\ Q \ \overset{\textbf{RC3}}{=}\ (x)\overline{a}x.\ [x \neq z]\ Q$$

Assume $P \equiv \sum_i \varphi_i\, \alpha_i.\, P_i$ and $Q \equiv \sum_j \psi_j\, \alpha_j.\, Q_j$ where no α_i (resp. β_i) binds a name free in Q (resp. P). Then infer:

$$P|Q = \sum_i \varphi_i\, \alpha_i.\,(P_i|Q) + \sum_j \psi_j\, \alpha_j.\,(P|Q_j) + \sum_{\alpha_i\ opp\ \beta_j} \big[\varphi_i \wedge \psi_j \wedge (x_i{=}y_j)\big]\, \tau.\, R_{ij}$$

where α_i *opp* β_j, x_i, y_j and R_{ij} are defined as follows

1. α_i is $\overline{x}_i u$ and β_j is $y_j(v)$; then R_{ij} is $P_i|Q_j\{{}^{u}\!/\!_{v}\}$;
2. α_i is $\overline{x}_i(u)$ and β_j is $y_j(v)$; then R_{ij} is $(w)\,(P_i\{{}^{w}\!/\!_{u}\}|Q_j\{{}^{w}\!/\!_{v}\})$ for $w \notin \mathrm{fn}(P_i, Q_j)$;
3. the converse of (1);
4. the converse of (2).

Table 6: The expansion axiom.

7.2 Parallel Composition

The only π-calculus operator which we have not considered so far is *parallel composition*. Despite its theoretical significance, with an interleaving semantics it is the easiest operator to handle: usually an *expansion* axiom is invoked to reduce the parallel composition of two finite agents to the sum of parallel-free agents. In this axiom the occurrence of a communication is represented by the the *silent action* τ. Operationally, $\tau.\, P$ is an agent which can evolve to P without requiring interactions with the environment. In the axiomatisations, we can deal with the silent prefix in the same way we did for free outputs; hence all we need is the inference rule

IP3 if $P = Q$ then $\tau.\, P = \tau.\, Q$

The expansion theorem we present for π-calculus (Table 6) is valid in all four equivalences studied in this paper; thus it is similar, but not identical, to the one in [MPW92], which does not involve conditionals and which is only valid for bisimilarity. Both the use of this axiom and the proof of its soundness are standard. We refer to [MPW92] for the operational semantics of parallel composition and for more discussion on the format of the expansion axiom.

8 Conclusion

8.1 Summary

We have established complete axiomatisations of equivalences for some variants of the π-calculus; these are also applicable for other languages where value-passing is a basic construct and where logical tests can be made on identity of the values.

Our choice of axioms brings homogeneity to the structure of the axiom systems. We move from a "late" to an "early" system by adding the axiom **SP**, from a "non-deterministic" to a "deterministic" system by removing the axioms for sum, and from a bisimilarity to a congruence system by replacing the laws which evaluate conditionals with laws for manipulating them, and by strengthening the inference rule for input prefix. An overview is shown in in Figure 1. Here, as previously, $\mathcal{L}$ stands for "late", $\mathcal{E}$ for "early" (these are the two versions of bisimulation), $\mathcal{D}$ for "deterministic" (meaning the subcalculus of deterministic agents), $\mathcal{B}$ for "bisimilarity" and $\mathcal{C}$ for "congruence". We only list the algebraic laws necessary for completeness; the brackets around the asterisks mean that these are only needed in $\mathcal{DC}$. We want to stress that all laws are valid in all systems, with the exception of **IP** in the systems for bisimilarities and **C1** and **C2** in the systems for congruences. If restriction and parallel composition are included in the language, the axiomatisations are extended with the laws of the preceding section. For ease of reference, Tables 7 and 8 in the Appendix contain a catalogue of all named laws in the paper (with the exception of the expansion law which is not repeated). Note that these contain axioms as well as derived laws.

8.2 Related Work

One of our main contributions is the treatment of conditionals, so it is interesting to relate our results to other formalisms with an "`if` φ `then` P `else` Q" construct. Although this appears to be the universal notation in Algol-type programming languages, the three-keyword formulation is awkward for laws with many nested conditionals. Unfortunately there is no consensus on a more compact notation. McCarthy [McC63] and others write $\varphi \rightarrow P, Q$; Hoare [Hoa85, H+87] and others write $P \lhd \varphi \rhd Q$; Bloom and Tindell [BT83] write $\kappa(\varphi, P, Q)$; Guessarian and Meseguer [GM87] write $[\varphi, P, Q]$; Manes [Man85] writes $\mathbf{If}_{\varphi}(P, Q)$. Our own $\varphi\ P\ Q$ works well for the purposes of this paper, and its brevity hopefully conveys an impartiality about notational conventions.

Axiomatic treatments of the conditional date back to McCarthy [McC63], where a functional language is considered as a universal programming notation. One of McCarthy's aims is to establish a theory of equivalence for this language. He is only

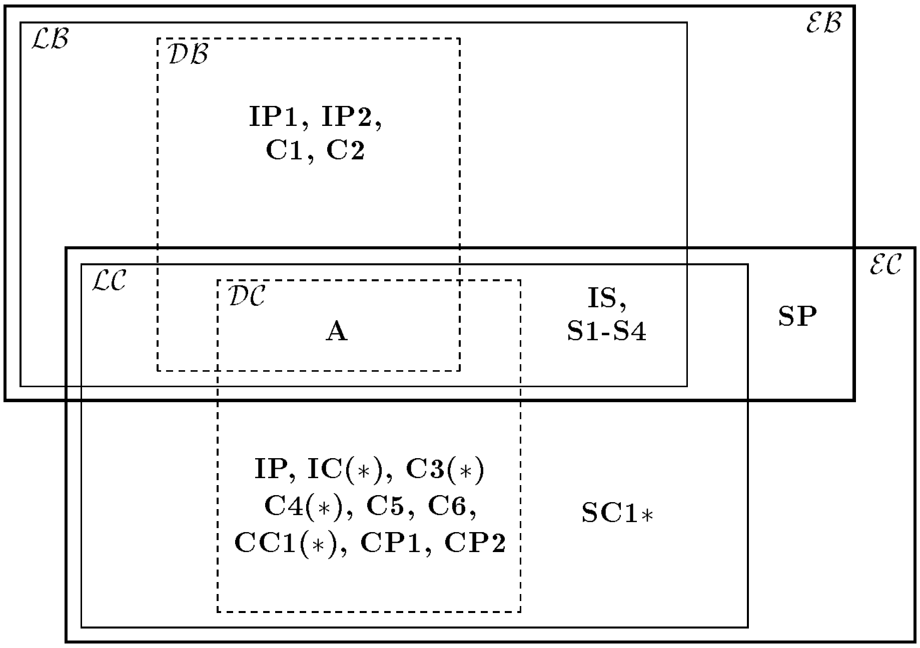

Figure 1: Overview of the axiom systems.

interested in functional aspects so there is no notion of "action"; in $\varphi \rightarrow P, Q$ all of φ, P and Q are again conditional forms or uninterpreted variables. The equality under under all interpretations of the variables is reminiscent of our theory of the congruences (which is equality under all substitutions), but McCarthy's axioms are significantly more complex, mainly because his conditions do not use the boolean connectives (disjunction, conjunction, negation). This line of work has been continued by Bloom and Tindell [BT83] who study the effects of adjoining the conditional construct to an arbitrary one-sorted algebra in various ways (one way is to let conditions consist of equality tests between elements of the algebra). None of the resulting classes of algebras is an equational class, although equality is completely axiomatised with laws similar to McCarthy's. Guessarian and Mesguer [GM87] extend these results to many-sorted and continuous algebras. Manes [Man85] explores the equational theory of conditionals over an arbitrary boolean algebra. Here the boolean conjunction and disjunction facilitate more compact axioms — there are some laws remarkably similar to our **C** and **CC** laws, although the underlying formalism is quite different.

In the setting of process algebra, value-passing and conditional constructs have been present since Milner's first presentation of CCS [Mil80]. Many of the now well-known axioms are given there but there are no completeness results, and the only laws for conditionals are counterparts of our **C1** and **C2**. In later publications, notably [Mil89], value-passing and conditionals are derived constructs obtained by translating input prefixes to sums over infinite sets of agents, and completeness results are given only for syntactically finite agents. This discrepancy was one of the reasons for the subsequent development of the π-calculus [MPW92] where value-passing and a restricted form of conditional are primitives (the influence of the π-calculus on our systems has been explained in the introduction). Another pioneering work in giving algebraic laws for conditionals in a reactive setting is Hoare's CSP [Hoa85], particularly as presented in [H+87]. The laws are inspired by McCarthy's. The completeness result is not comparable to our results since it is a form of relative completeness and the semantics are different (programs operate on a store).

More closely related to our work is a series of papers by Hennessy and coauthors. In a process algebra, Hennessy and Ingólfsdóttir [HI89] made a deep analysis of a language where value-passing is a primitive. A denotational model and axiomatisation were given where the equivalence and preorders are based on testing. The axiomatisation contains a counterpart to our law **IP2**; the only laws for conditionals are the counterparts of **C1** and **C2**. Boreale and DeNicola [BD92] and Hennessy [He91] apply a similar system to an axiomatisation of testing equivalence in the π-calculus. Hennessy explores an alternative proof system in [He91b]. This system explicitly represents assumptions about the values of free value variables, and there is a "cut" rule to split such assumptions (thus **IP2** is not needed since the cut rule can accomplish any necessary case analysis). Hennessy and Lin [HL93] adapt this system to late and early bisimulation semantics. The resulting systems are more complex and more powerful than our $\mathcal{LB}$ and $\mathcal{EB}$, since they incorporate arbitrary tests (not only matching) and expressions with value variables, and carefully separate proofs about value expressions from proofs about agents. In contrast, our concern is to obtain small and clear proof systems for more restricted languages. This difference of intent is obvious e.g. in the axioms which distinguish late and early bisimilarity. We use **SP**, whereas Hennessy and Lin use:

$$\frac{\sum_{i\in I} \tau.\, P_i = \sum_{j\in J} \tau.\, Q_j}{\sum_{i\in I} a(x).\, P_i = \sum_{j\in J} a(x).\, Q_j}$$

Note that the premise may contain a free x which is bound in the conclusion; the premise thus has an implicitly universally quantified x.

By the time this paper has been completed, one of us [San93] has studied and axiomatised a variant of late bisimilarity in which, intuitively, the instantiation of the parameter of an input happens only when the name is needed. An attractive feature of this bisimilarity is that it is also a congruence. An accurate comparison between the axiomatisations here and in [San93] is difficult, due to the difference in the conditional constructs used: In [San93] only matching is employed and the generalisation of the theory to a calculus with mismatching is not obvious.

Other efforts in process algebra are more remotely related to the present work. We would here mention Groote and Ponse [GP90] who introduce "guards" as a primitive conditional construct. The setting is different since they do not consider value-passing, but they formulate a law (their "**G4**") remarkably similar to our **SP**. A later work by the same authors [GP91] defines a specification language with value-passing and conditionals built on top of ACP in a modular way; the system is akin to our axiomatisation of bisimilarity in that the laws for conditionals imply evaluation of the conditions. A similar language is investigated by Mauw [Mauw91].

8.3 Further Work

One idea for further work is to clarify the role of mismatching. In the original π-calculus syntax this is absent; there is only the unary matching operator. There is a good reason for this: the π-calculus already possesses a "matching" ability in the sense that equality of port names in different parallel components may enable an interaction between the components. So the addition of matching causes no dramatic change of the theory; in fact most occurrences of matching can be encoded by parallel composition.

As pointed out in [Par90], where mismatching in the π-calculus and the law **SP** were first considered, the addition of mismatching preserves the theory of the calculus with the minor exception of the following monotonicity property: substitutions may only increase the action capabilities of an agent:

$$\text{if } P \xrightarrow{\alpha} P' \text{ then also } P\sigma \xrightarrow{\alpha\sigma} P'\sigma$$

(By alpha-conversion we assume that the bound names of P are not mentioned in σ.) With mismatching, this property holds for *injective* substitutions σ (see Lemma 4.4) but not in general. For instance, $[x \neq y]\alpha \xrightarrow{\alpha}$ but not $([x \neq y]\alpha)\{x/y\} \xrightarrow{\alpha}$. This failure shows that mismatching cannot be encoded in the original π-calculus.

In the axiomatisations mismatching plays a central role: Of the four equivalences considered in this paper, mismatching is superfluous only for late bisimilarity whereas it appears to be necessary for the remaining three, if we want to keep the laws reasonably simple. For instance, let α be an output action and α^n the sequence of n α-actions, and consider

$$P_n \equiv \alpha^n.\beta + \alpha^n \qquad Q_n \equiv P_n + \alpha^n.[x = y]\beta$$

For any integer n, the agents P_n and Q_n are (both early and late) congruent, but their equalities appear difficult to recover without mismatching. (A similar example can be given using parallel composition in place of matching.) It may be interesting to see if the necessity of mismatching can be established formally. Another avenue of research is to determine precisely how mismatching increases the expressiveness of the calculus.

Our axiomatisations leave open the question of determining canonical representatives for the equivalence classes of the behavioural equivalences. In CCS (or in other traditional process algebras) the axiomatisation of the bisimilarities provide a guideline for the construction of canonical representatives through the notion of normal form (obtained from a head normal form by normalising all its subcomponents). Our axiomatisations, both for the bisimilarities and for the congruences, also use the notion of head normal form. However, with the bisimilarities the head normal forms do not lead to a definition of normal form: The problem is with the normalisation of an agent P underneath an input prefix, since the rule **IP2** for input prefix has a set of multiple and non-disjoint premises involving P. In contrast, with the congruences a head normal form can be transformed into a normal form but now the problem is with their unicity: The head normal forms depend upon the set of free names of the agents, but equivalent agents might have different sets of free names.

The algorithmic aspects of equality in value-passing languages are largely unexplored. Hennessy and Lin [HL92] provide an algorithm, based on symbolic execution, for some interesting subclasses of the language of [HL93]. Our completeness result implies an algorithm, but because of the saturation in the normal forms this immediately generates an exponential blow-up. If conditions with boolean connectives are admitted it is of course NP-hard to determine if even a simple equation such as $\varphi P = \psi P$ holds, but if (as in the π-calculus) only elementary conditionals are allowed more efficient algorithms may be possible. Sethi [Set78] gives a polynomial algorithm to determine if nested elementary conditional forms are semantically equal and it would be interesting to see if this idea can be transferred to the π-calculus. If recursion is added to our language with prefixing and sum, the equivalence problem is NP-hard even if no conditions at all are allowed [JP89].

Finally it should be possible to extend our results to other equivalences. A good first candidate is the weak bisimulation equivalence, or "observation" equivalence. As for strong bisimilarity this proliferates into an early and a late version, and fails to be preserved by input prefix. Although we expect that the axiomatisations are recovered by adding the three "τ-laws" of [Mil89] (as is done in [HL93]) to our systems this remains to be worked out.

References

[BK85] J.A. Bergstra and J.W. Klop. Algebra of Communicating Processes with Abstraction. *Theoretical Computer Science* 33:77-121 (1985).

[BB90] G. Berry and G. Boudol. The Chemical Abstract Machine. *Theoretical Computer Science* 96:217–248 (1992).

[BT83] S. Bloom and R. Tindell. Varieties of "if-then-else". *SIAM J. Computing* 12(4):677–707 (1983).

[BD92] M. Boreale and R. DeNicola. Testing equivalence for mobile processes. In Cleaveland (Ed): *Proceedings of CONCUR '92*, Stony Brook, August 1992, pages 2–16, Springer Verlag LNCS 630 (1992).

[GP90] J. Groote and A. Ponse. Process algebra with guards. CWI technical report CS-R9069, Amsterdam 1990. To appear in *Formal Aspects of Computing.*

[GP91] J. Groote and A. Ponse. Proof Theory for μCRL. CWI technical report CS-R9138, Amsterdam 1991.

[GM87] I. Guessarian and J. Meseguer. On the axiomatization of "if-then-else". *SIAM J. Computing* 16(2):322–357 (1987).

[He91] M. Hennessy. A Model for the π-calculus, Tech. Report 91/08, Department of Computer Science, University of Sussex, 1991.

[He91b] M. Hennessy. A proof system for communicating processes with value-passing. *Formal Aspects of Computing* 3:346–366 (1991).

[HI89] M. Hennessy and A. Ingólfsdóttir. A theory of communicating processes with value-passing. Technical Report 3/89, Univ. of Sussex 1989. To appear in *Information and Computation.*

[HL93] M. Hennessy and H. Lin. Proof systems for message-passing algebras. Draft, University of Sussex 1993.

[HL92] M. Hennessy and H. Lin. Symbolic bisimulations. Technical Report, University of Sussex 1992.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[H+87] C.A.R. Hoare, I. Hayes, He Jifeng, C. Morgan, A. Roscoe, J. Sanders, I. Sorensen, J. Spivey and B. Sufrin. Laws of programming. *Comm. of the ACM* 30(8):672–686 (1987).

[JP89] B. Jonsson and J. Parrow. Deciding bisimulation equivalences for a class of non-finite-state programs. In Monien, Cori (Eds): *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science*, February 1989, pages 421-433. Springer Verlag LNCS 349, 1989. Accepted for publication in *Information and Computation.*

[Man85] E. Manes. Guard modules. *Algebra Universalis* 21:103–110 (1985).

[Mauw91] S. Mauw. *PSF — A Process Specification Formalism.* Ph. D. Thesis, University of Amsterdam 1991.

[McC63] J. McCarthy. A basis for a mathematical theory of computation. In Braffort, Hirschberg (Eds): *Computer Programming and Formal Systems*, pages 33–70, North-Holland (1963).

[Mil80] R. Milner. *A Calculus of Communicating Systems.* Springer Verlag LNCS 92, 1980.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[Mil92] R. Milner. Functions as Processes. *J. of Mathem. Structures in Computer Science* 2(2):119–141 (1992).

[MPW91] R. Milner, J. Parrow and D. Walker. Modal logics for mobile processes. In Baeten, Groote (Eds): *Proceedings of CONCUR '91*, Amsterdam, August 1991, pages 45–60, Springer Verlag LNCS 527 (1991). To appear in *Theoretical Computer Science.*

[MPW92] R. Milner, J. Parrow and D. Walker. A Calculus of Mobile Processes, Part I and II. *Information and Computation* 100:1–77 (1992).

[MS92] R. Milner and D. Sangiorgi. Barbed Bisimulation, In Kuich, W. Ed: *Proceedings of ICALP '92*, pages 685–695, Springer Verlag LNCS 623, (1992).

[Mol89] F. Moller. *Axioms for concurrency.* PhD thesis, CST-59-89, Department of Computer Science, University of Edinburgh, 1989.

[Par90] J. Parrow. 'Mismatching' and early equivalence (π-calculus note JP13). Manuscript, Swedish Institute of Computer Science 1990.

[San92] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* PhD thesis CST–99–93, Department of Computer Science, University of Edinburgh, 1992.

[San93] D. Sangiorgi. A theory of bisimulation for π-calculus. Technical Report ECS–LFCS–93–270, LFCS, Department of Computer Science, University of Edinburgh, 1993. Extended Abstract in *Proc. CONCUR '93*, Springer Verlag LNCS 715 (1993).

[Set78] R. Sethi. Conditional expressions with equality tests. *J. ACM* 25(4):667–674 (1978).

Appendix: List of Laws

IP1	If $P = Q$ then $\overline{a}x.\,P = \overline{a}x.\,Q$	
IP2	If $P\{y/x\} = Q\{y/x\}$ for all $y \in \mathrm{fn}(P, Q, x)$ then $a(x).\,P = a(x).\,Q$	
IP	If $P = Q$ then $\alpha.\,P = \alpha.\,Q$	
IS	If $P = Q$ then $P + R = Q + R$	
IC	If $P = Q$ then $\varphi\,P = \varphi\,Q$	
IC$*$	If $P = Q$ then $\varphi\,P\;R = \varphi\,Q\;R$	
IR	If $P = Q$ then $(x)\,P = (x)\,Q$	
S1	$P + \mathbf{0} = P$	
S2	$P + P = P$	
S3	$P + Q = Q + P$	
S4	$P + (Q + R) = (P + Q) + R$	
C1	$\varphi\,P\;Q = P$	if $[\![\varphi]\!] = \mathit{True}$
C2	$\varphi\,P\;Q = Q$	if $[\![\varphi]\!] = \mathit{False}$
C3	If $\varphi \Leftrightarrow \psi$ then $\varphi\,P = \psi\,P$	
C3$*$	If $\varphi \Leftrightarrow \psi$ then $\varphi\,P\;Q = \psi\,P\;Q$	
C4	$\mathit{False}\;P = \mathit{False}\;Q$	
C4$*$	$\mathit{False}\;Q\;P = \mathit{False}\;R\;P$	
C5	$\varphi\,P\;P = P$	
C6	$\varphi\,P\;Q = \neg\varphi\,Q\;P$	
C7	$\mathit{True}\;P\;Q = P$	

Table 7: The algebraic laws, I.

CC1	$\varphi\,(\,\psi\, P\,) = [\varphi \wedge \psi]\, P$	
CC1∗	$\varphi\,(\,\psi\, P\, Q\,)\, R = [\varphi \wedge \psi]\, P\,(\,\varphi\, Q\, R\,)$	
CC2	$\varphi\, P\,(\,\psi\, P) = [\varphi \vee \psi]\, P$	
CC3	$\varphi\,(\,\varphi\, P)\, Q = \varphi\, P\, Q$	
CC4	$\varphi\, P\,(\,\psi\, Q\, R\,) = \psi\, Q\,(\,\varphi\, P\, R\,)$	if $\psi \wedge \varphi \Leftrightarrow \mathit{False}$
CC5	$[x = y]\,(\,[x \neq y]\, P) = \mathbf{0}$	
SC1	$\varphi\,(P + Q) = \varphi\, P + \varphi\, Q$	
SC1∗	$\varphi\,(P_1 + P_2)\,(Q_1 + Q_2) = \varphi\, P_1\, Q_1 + \varphi\, P_2\, Q_2$	
SC2	$\varphi\, P\, Q = \varphi\, P + \neg\varphi\, Q$	
SC3	$P = \varphi\, P + \neg\varphi\, P$	
SC4	$[\varphi \vee \psi]\, P = \varphi\, P + \psi\, P$	
CP1	$\varphi\,(\alpha.\, P) = \varphi\,(\alpha.\, \varphi\, P)$	if $\mathrm{bn}(\alpha) \cap \mathrm{n}(\varphi) = \emptyset$
CP2	$[x = y]\, \alpha.\, P = [x = y]\,(\alpha\{x/y\}).\, P$	
SP	$a(x).\, P + a(x).\, Q = a(x).\, P + a(x).\, Q + a(x).\, \Big([x = y]\; P\; Q\Big)$	
R	$(x)\,\mathbf{0} = \mathbf{0}$	
RR	$(x)\,(y)\,P = (y)\,(x)\,P$	
RS	$(x)\,(P + Q) = (x)\,P + (x)\,Q$	
RP1	$(x)\,\alpha.\,P = \alpha.\,(x)\,P$	if $x \notin \mathrm{n}(\alpha)$
RP2	$(x)\,\alpha.\,P = \mathbf{0}$	if x is the port of α
RC1	$(x)\,[x = y]\,P = \mathbf{0}$	if $x \neq y$
RC2	$(x)\,[z = y]\,P = [z = y]\,(x)P$	if $x \neq y, z$
RC3	$(x)\,[x \neq y]\,P = (x)P$	if $x \neq y$
RC4	$(x)\,[y \neq z]\,P = [y \neq z]\,(x)P$	if $x \neq y, z$
RC5	$(x)\,(\,\varphi\, P\, Q\,) = \mathit{Remove}_x(\varphi)\;(x)\,P\;(x)\,Q$	

Table 8: The algebraic laws, II.