Bisimulations for asynchronous mobile processes

Martin Hansen Hans Hüttel Josva Kleist

BRICS*
Department of Computer Science
University of Aalborg
Fredrik Bajersvej 7E, 9220 Aalborg Ø, Denmark

Abstract

Within the past few years there has been renewed interest in the study of value-passing process calculi as a consequence of the emergence of the π-calculus. Here, [MPW89] have determined two variants of the notion of *bisimulation*, *late* and *early* bisimilarity. Most recently [San93] has proposed the new notion of *open* bisimulation equivalence.

In this paper we consider Plain LAL, a mobile process calculus which differs from the π-calculus in the sense that the communication of data values happens *asynchronously*. The surprising result is that in the presence of asynchrony, the open, late and early bisimulation equivalences *coincide* – this in contrast to the π–calculus where they are distinct. The result allows us to formulate a common *equational theory* which is sound and complete for finite terms of Plain LAL.

1 Introduction

An important question in the theory of process calculi is when two processes can be said to exhibit the same behaviour and a number of *behavioural equivalences* have been proposed. Traditionally, the emphasis has been on studying the equivalences for 'pure' processes where communication does not involve the passing of data values. However, within the past few years there has been renewed interest in the study of value-passing as a consequence of the emergence of the π-calculus. Here [MPW89] have determined two

*Basic Research in Computer Science,
Centre of the Danish National Research Foundation.

variants of the notion of *bisimulation*, *late* and *early* bisimilarity, differing in the assumption of when a data value is instantiated. Most recently [San93] has proposed a new notion of bisimulation equivalence which he calls *open* bisimilarity. Sangiorgi formulates a sound and complete *equational theory* for open bisimilarity on finite (i.e. non-recursive) π-calculus terms.

An important theorem is that the three notions of bisimilarity are distinct within the π-calculus, with open bisimulation being strictly finer than late bisimulation which in turn is strictly finer than early bisimulation. The late/early distinction has been studied for the much coarser *testing* equivalence of Hennessy and De Nicola and it has been shown by [Ing94] and [BN92] that the late and early testing equivalences coincide, both for a process calculus with simple data values and conditional expression and for the π-calculus.

In this paper we consider Plain LAL, a mobile process calculus which differs from the π-calculus in the sense that the communication of data values happens *asynchronously*. Previously, [HT91] and [Bou92] have described so-called asynchronous π-calculi. In [Ode95] it is shown how familiar programming constructs can be encoded in a straightforward fashion within an asynchronous π-calculus, thus confirming the naturalness of the combination of asynchrony and mobility.

The main result of the present paper is that in the presence of asynchrony, the open, late and early bisimulation equivalences *coincide*. A related results was established by [HY93]; however this result was shown for another asynchronous calculus, the simpler so-called γ-calculus. As the equivalences are one and the same in Plain LAL, this leads us to formulate an *equational theory* which is sound for finite terms of Plain LAL. Most proofs of theorems have been omitted; they can be found in [HK94].

2 Plain LAL

The process calculus that we shall consider here uses a notion of asynchrony inspired by the Linda paradigm for parallel programming proposed by [GB82]. In Linda, a program is viewed as a collection of concurrent processes and data ('tuples') existing in a common 'tuple space'. Tuples communicate by depositing tuples in the 'tuple space' by using the out and eval operators and retrieving them using the rd and in operators. When retrieving another tuple, a Linda tuple can specify which kinds of tuples that may be retrieved by means of a notion of pattern matching.

Plain LAL is a two-level calculus as we distinguish between processes and systems. Processes correspond to tuples in a Linda 'tuple space' and are deposited by a *spawn* operator at the level of processes. This is the only means of introducing parallelism; Plain LAL has no explicit parallel composition operator for processes and is in this respect similar not only to out and eval operators of Linda, but also to the fork calculus of [Hav94] and to notions underlying the semantics of applicative languages with concurrency, such as Concurrent ML ([BMT92]).

The pattern matching of Linda used to retrieve tuples is modelled by Plain LAL processes having *names*. A Plain LAL *agent* (or *system*) $S, T, \ldots \in$ **Agent** is the parallel composition of a set of named processes, each of the form (α, P) where $\alpha, \beta, \gamma \ldots \in$ **Name** range over *names* and $P, Q, \ldots \in$ **Proc** range over *processes*. The only data that can be communicated in Plain LAL are names; a process (α, P) can obtain the name γ from another process (β, γ) via the name β, called the *location* of γ.

2.1 Syntax

Expressions in Plain LAL are given by the following syntax:

$$\begin{array}{rcl} S & ::= & (\alpha, P) \mid (\alpha)S \mid S_1|S_2 \mid \bullet \\ P & ::= & \sum\limits_{i \in I} a_i.P_i \mid (\alpha)P \mid \alpha \mid \xi(P) \\ a & ::= & \alpha!P \mid \alpha?\beta \mid \tau \end{array}$$

The chief operators of Plain LAL are the prefixing operators. In Plain LAL, the 'output' prefix $\alpha!P$ assumes the role of the spawn operator – it creates a parallel component (α, P), i.e. a process P located at α. The 'input' prefix $\alpha?\beta$ asks for a name from any parallel component whose location is α and binds the name to β. The τ prefix denotes silent moves.

Names can be made local using the *restriction* operator; $(\alpha)S$ denotes that α is a name local to S. We define $(L)P$ resp. $(L)S$ where $L \subseteq$ **Name** as P resp. S restricted by all names in L. Input prefixing and restriction work as binding operators on names and give rise to the familiar definitions of free and bound names.

Processes can be built using sums of prefixed processes, restriction and *values* α. $\xi(P)$ is the replication operator of the π-calculus; we need this to be able to express infinite behaviours (alternatively, we could have added a recursion operator to the syntax; the two notions are inter-expressible.)

Systems are the parallel composition of processes and can themselves be composed in parallel. We let $\Pi_{i\in I}S_i$ denote the parallel composition of a number of systems where I is an index set. In the rest of this paper, we shall assume that any such index set is finite; we sometimes omit the index set and indices when they are clear from the context. We write $\mathbf{0}$ for $\sum_{\emptyset} a.P$ and a for $a.\mathbf{0}$; further we use $+$ for binary sum.

2.2 Semantics

Our operational semantics for Plain LAL consists of three levels: The basis is a *commitment semantics* for processes which describes which actions processes can commit to. At the system level we have a *labelled transition system* which, based on the commitments of the processes, controls which interactions can occur. To simplify the rules we have a *structural congruence* which lets us manipulate agents by their structure.

The commitment semantics for processes is given by:

$$
\begin{array}{ll}
[\text{action}_{\succ}] & a.P \succ a.P \\
[\text{sum}_{\succ}] & \dfrac{P_j \succ a.P}{\sum_I P_i \succ a.P} \text{ if } j \in I \\
[\text{res}_{\succ}] & \dfrac{P \succ a.P'}{(\alpha)P \succ a.(\alpha)P'} \text{ if } \alpha \notin \mathsf{fn}(a)
\end{array}
$$

The structural congruence $\equiv$ is defined as the smallest congruence relation satisfying:

$$
\begin{array}{rcl}
S & \equiv & T \text{ if } S \text{ and } T \text{ are } \alpha\text{-convertible} \\
(\alpha)(\beta,\mathbf{0}) & \equiv & (\beta,\mathbf{0}) \\
(\alpha,\mathbf{0}) & \equiv & \bullet \\
(\beta)(\alpha)S & \equiv & (\alpha)(\beta)S \\
(\alpha)((\beta,P)|(\gamma,Q)) & \equiv & (\alpha)(\beta,P)|(\gamma,Q) \text{ if } \alpha \notin \mathsf{fn}(Q) \cup \{\gamma\} \\
(\alpha,(\beta)P) & \equiv & (\beta)(\alpha,P) \text{ if } \alpha \neq \beta \\
(\alpha,\xi(P)) & \equiv & (\alpha.P)|(\alpha,\xi(P))
\end{array}
$$

$(S/\equiv, |, \bullet)$ is a symmetric monoid

Finally, the labelled transition system takes labels $m \in \mathbf{Label}$ given by the following syntax:

$$[\text{input}] \quad \frac{P \succ \alpha?\beta.P'}{(\alpha, P) \xrightarrow{\alpha?\beta} (\alpha, P')}$$

$$[\text{value}] \quad (\alpha, \beta) \xrightarrow{\beta@\alpha} \bullet$$

$$[\text{open}] \quad \frac{S \xrightarrow{\beta@\alpha} S'}{(\beta)S \xrightarrow{(\beta)@\alpha} S'}$$

$$[\text{com}_1] \quad \frac{S \xrightarrow{\beta@\alpha} S' \qquad T \xrightarrow{\alpha?\gamma} T'}{S|T \xrightarrow{\tau} S'|T'\{\beta/\alpha\}}$$

$$[\text{com}_2] \quad \frac{S \xrightarrow{(\beta)@\alpha} S' \qquad T \xrightarrow{\alpha?\gamma} T'}{S|T \xrightarrow{\tau} (\beta)(S'|T'\{\beta/\alpha\})} \text{ if } \beta \notin \mathsf{fn}(T)$$

$$[\text{spawn}] \quad \frac{P \succ \alpha!Q.P'}{(\beta, P) \xrightarrow{\tau} (\beta, P')|(\alpha, Q)}$$

$$[\text{internal}] \quad \frac{P \succ \tau.P'}{(\alpha, P) \xrightarrow{\tau} (\alpha, P')}$$

$$[\text{par}] \quad \frac{S \xrightarrow{m} S'}{S|T \xrightarrow{m} S'|T} \text{ if } \mathsf{bn}(m) \cap \mathsf{fn}(T) = \emptyset$$

$$[\text{res}] \quad \frac{S \xrightarrow{m} S'}{(\alpha)S \xrightarrow{m} (\alpha)S'} \text{ if } \alpha \notin \mathsf{fn}(m)$$

$$[\text{struct}] \quad \frac{S \equiv S' \quad S' \xrightarrow{m} T' \quad T' \equiv T}{S \xrightarrow{m} T}$$

Table 1: The labelled transition semantics of Plain LAL

$$m ::= \tau \mid \alpha?\beta \mid \alpha@\beta \mid (\alpha)@\beta$$

The first two indicate internal and input actions respectively, the third means that the agent contains the free name α at the free name β and the last means that the agent contains the bound name α at the free name β. The transition relation $\longrightarrow \subseteq (\textbf{Agent} \times \textbf{Label} \times \textbf{Agent})$ is the smallest relation satisfying the rules in Table 1.

3 Bisimulations

For value passing calculi several formulations of bisimulations have been proposed, among these early, late and open bisimulation. In this section we shall define these for Plain LAL and study their relationship.

Though quite similar in definition they all differ in the case of the π-calculus. The main result of this article is that these in fact coincide in the case of Plain LAL.

The notion of early bisimulation was first defined in [MPW89]; the idea is that a transition can be matched in different ways, depending on the value that is being communicated. In our setting we have:

Definition 1 *Early bisimulation A relation $\mathcal{R}$ is an early bisimulation if it is symmetric and $S \mathrel{\mathcal{R}} T$ implies*

- *If $S \xrightarrow{\alpha?\gamma} S'$ then for every name $\beta \in$ **Name** there exists a T' such that $T \xrightarrow{\alpha?\gamma} T'$ and $S'\{\beta/\gamma\} \mathrel{\mathcal{R}} T'\{\beta/\gamma\}$.*
- *If $S \xrightarrow{m} S'$ and $m \in \{\tau, \beta@\alpha, (\beta)@\alpha\}$ then there exists a T' s.t. $T \xrightarrow{m} T'$ and $S' \mathrel{\mathcal{R}} T'$.*

Two programs S and T are early bisimilar, written $S \sim T$, if $S \mathrel{\mathcal{R}} T$ for some early bisimulation $\mathcal{R}$.

Late bisimulation was also defined in [MPW89]. Intuitively, the difference is that a transition must be matched by the *same* move, irrespective of the value communicated. In Plain LAL, we get:

Definition 2 (Late bisimulation) *A relation $\mathcal{R}$ is a late bisimulation if it is symmetric and $S \mathrel{\mathcal{R}} T$ implies*

- *If $S \xrightarrow{\alpha?\gamma} S'$ then there exists a T' such that $T \xrightarrow{\alpha?\gamma} T'$ and for every name β $S'\{\beta/\gamma\} \mathrel{\mathcal{R}} T'\{\beta/\gamma\}$.*
- *If $S \xrightarrow{m} S'$ and $m \in \{\tau, \beta@\alpha, (\beta)@\alpha\}$ then there exists a T' s.t. $T \xrightarrow{m} T'$ and $S' \mathrel{\mathcal{R}} T'$.*

Two programs S and T are late bisimilar, written $S \sim_L T$, if $S \mathrel{\mathcal{R}} T$ for some late bisimulation $\mathcal{R}$.

Finally, open bisimulation, a notion of equivalence first studied by [San93]. The underlying idea is that one should consider matches for all possible instantiations of variables.

Definition 3 (Open bisimulation) *A relation $\mathcal{R}$ is an open bisimulation if it is symmetric and $S\,\mathcal{R}\,T$ implies that for all substitutions $\vartheta \in$ **Name** $\rightarrow$ **Name** it holds that if $S\vartheta \xrightarrow{m} S'$ then there exists a T' such that $T\vartheta \xrightarrow{m} T'$ and $S'\,\mathcal{R}\,T'$.*

Two programs S and T are open bisimilar, written $S \sim_O T$, if $S\,\mathcal{R}\,T$ for some open bisimulation $\mathcal{R}$.

By inspecting the definitions, it is easy to see that $\sim_O \subseteq \sim_L \subseteq \sim$. Further, it is relatively easy to verify that all three are indeed equivalence relations and also that they are congruence relations with respect to all operators but prefixing. In fact they are also congruences with respect to prefixing; we shall return to this later.

To prove our results in the following we shall exploit the techniques of up-to bisimulations. It is quite simple to prove that this principle is also sound for Plain LAL.

We shall now investigate how these 3 definitions of bisimulation relate to each other. But first we need the following lemma which states an important difference between synchronous and asynchronous communication

Lemma 1 *S has a τ-transition, $S \xrightarrow{\tau} T$, due to an internal communication on a free name α iff either*

- $S \xrightarrow{\beta @ \alpha} S' \xrightarrow{\alpha ? \gamma} S''$ *with* $T \equiv S''\{\beta/\gamma\}$
- $S \xrightarrow{(\beta) @ \alpha} S' \xrightarrow{\alpha ? \gamma} S''$ *with* $T \equiv (\beta) S''\{\beta/\gamma\}$

Proof. We have either $S \equiv S' | (\alpha, \beta)$ or $S \equiv (\beta)(S' | (\alpha, \beta))$. $\square$

Observe that the *if* part of the above lemma does not hold in a synchronous calculus because in a synchronous calculus a sequential process cannot communicate with itself. For instance we have $\bar{\alpha}\beta.\alpha(\gamma) \xrightarrow{\bar{\alpha}\beta} \alpha(\gamma) \xrightarrow{\alpha(\gamma)} \mathbf{0}$ in the π-calculus but clearly we cannot have internal communication on α.

Using Lemma 1 we can now state the following theorem which turns out to be extremely important.

Theorem 2 *If $S \sim T$ then $S\{\alpha/\gamma\} \sim T\{\alpha/\gamma\}$ for all $\alpha \in$ **Name**.*

Proof. We show that $\mathcal{R} = \{(S\{\alpha/\gamma\}, T\{\alpha/\gamma\}) \mid S \sim T\}$ is an early bisimulation up to structural congruence.

The proof proceeds by inspecting the possible transitions from $S\{\alpha/\gamma\}$. We only consider two cases:

$S\{\alpha/\gamma\} \xrightarrow{\beta?\delta} S''$: Then either $S \xrightarrow{\beta?\delta}$ or $S \xrightarrow{\gamma?\delta}$ with $\beta = \alpha$.

For $S \xrightarrow{\beta?\delta} S'$ we have $S'' = S'\{\alpha/\gamma\}$. Since $S \sim T$ we have for all $\eta \in \mathbf{Name}$ that $T \xrightarrow{\beta?\delta} T'$ such that $S'\{\eta/\delta\} \sim T'\{\eta/\delta\}$. Further we must have $T\{\alpha/\gamma\} \xrightarrow{\beta?\delta} T'\{\alpha/\gamma\}$ since $\gamma \neq \beta$. Now for all η we have $S'\{\alpha/\gamma\}\{\eta/\delta\} \ \mathcal{R} \ T'\{\alpha/\gamma\}\{\eta/\delta\}$ where we w.l.o.g. assume that $\delta \neq \alpha$, $\delta \neq \gamma$ and $\eta \neq \gamma$.

For $S \xrightarrow{\gamma?\delta} S'$ we have $S'' = S'\{\alpha/\gamma\}$. Since $S \sim T$ we have for all $\eta \in \mathbf{Name}$ that $T \xrightarrow{\gamma?\delta} T'$ such that $S'\{\eta/\delta\} \sim T'\{\eta/\delta\}$. Hence $T\{\alpha/\gamma\} \xrightarrow{\alpha?\delta} T'\{\alpha/\gamma\}$ is a matching move.

$S\{\alpha/\gamma\} \xrightarrow{\tau} V$: Then either $S \xrightarrow{\tau} S'$ with $V = S'\{\alpha/\gamma\}$ in which case the proof is similar to the above, or the transition is due to internal communication on α. We examine the latter case.

Since α is free we have by Lemma 1 one of two:

i. $S\{\alpha/\gamma\} \xrightarrow{\beta@\alpha} V' \xrightarrow{\alpha?\delta} V''$ and $V \equiv V''\{\alpha/\delta\}$.

ii. $S\{\alpha/\gamma\} \xrightarrow{(\beta)@\alpha} V' \xrightarrow{\alpha?\delta} V''$ and $V \equiv (\beta)V''\{\beta/\delta\}$.

We shall only consider case ii. Assume w.l.o.g. that $\beta \neq \gamma$, $\delta \neq \gamma$ and $\alpha \neq \delta$. It is fairly obvious that either $S \xrightarrow{(\beta)@\alpha} S' \xrightarrow{\gamma?\delta} S''$ or $S \xrightarrow{(\beta)@\gamma} S' \xrightarrow{\alpha?\delta} S''$ with $S'\{\alpha/\gamma\} = V'$ and $S''\{\alpha/\gamma\} = V''$ since S could not do the transition. We assume w.l.o.g. that it is the second case. Then $T \xrightarrow{(\beta)@\gamma} T' \xrightarrow{\alpha?\delta} T''$ with $S' \sim T'$ and $S''\{\beta/\delta\} \sim T''\{\beta/\delta\}$. Further we must have $T\{\alpha/\gamma\} \xrightarrow{(\beta)@\alpha} T'\{\alpha/\gamma\} \xrightarrow{\alpha?\delta} T''\{\alpha/\gamma\}$ and hence by Lemma 1 $T\{\alpha/\gamma\} \xrightarrow{\tau} (\beta)T''\{\alpha/\gamma\}\{\beta/\delta\}$[1]. Now we have

$$V \equiv (\beta)S''\{\alpha/\gamma\}\{\beta/\delta\} = ((\beta)S''\{\beta/\delta\})\{\alpha/\gamma\} \ \mathcal{R} \ (\beta)T''\{\alpha/\gamma\}\{\beta/\delta\}$$

A symmetric argument exists for $T\{\alpha/\gamma\} \xrightarrow{m} V$. □

As an immediate corollary we have that all our definitions of bisimulation coincide.

Corollary 3 *Early, late and open bisimilarity coincide.*

[1]This does *not* hold in the π-calculus.

Proof. Early bisimilarity implies open because by Theorem 2 we know that if $S \sim T$ then for every substitution ϑ we have $S\vartheta \sim T\vartheta$. Therefore we can drop the substitution in the input case of the definition of early bisimulation. We now have that for every transition $S\vartheta \xrightarrow{m} S'$ there exists a T' such that $T\vartheta \xrightarrow{m} T'$ with $S' \sim T'$, hence $\sim$ is an open bisimulation. $\square$

This result is really interesting, as in the π-calculus, even without the match operator, early bisimulation is strictly coarser than late bisimulation which in turn is strictly coarser than open bisimulation. This is evidence that the expressive power of asynchronous communication is in fact less than that of synchronous communication.

A more pragmatic consequence of Corollary 3 is that it does not matter which definition of bisimilarity we use because we know that the underlying equivalence is the same.

Corollary 4 (Congruence) $\sim$ *is a congruence relation.*

Proof. Since $\sim$ is an open bisimulation it is clear that $\sim$ must be a congruence with respect to prefixing. $\square$

The result above does not hold for the π-calculus where neither early nor late bisimulation is a congruence.

As stated in the proof of Corollary 3 we have by Theorem 2 that if we omit the substitution in the input case of the definition of early bisimulation we do not get a coarser relation. However, the definition we end up with is the definition of ground bisimulation. Thus we have no less than 6 bisimulations which are different in the π-calculus and coincide in Plain LAL. Despite this, by closer inspection it turns out that the only essential limitation Plain LAL has to the π-calculus is a more limited type of summation, since it can be shown that all π-calculus operators except the general operator can be encoded in plain LAL ([HK94]).

3.1 Weak Bisimulation

The proofs in the proof above are not dependent on dealing with strong bisimulations and are easily generalized to weak bisimulation. Hence all weak versions of the above bisimulations (save early and late congruence) are equal to the weak bisimulation given by the following definition. We define a weak transition as

Definition 4 *The relation* $\Longrightarrow \subseteq (\textbf{Agent} \times \textbf{Label} \cup \{\varepsilon\} \times \textbf{Agent})$ *is given by:*

$$\overset{m}{\Longrightarrow} = \overset{\tau}{\longrightarrow}^{*} \overset{m}{\longrightarrow}$$

$$\overset{\varepsilon}{\Longrightarrow} = \overset{\tau}{\longrightarrow}^{*}$$

Definition 5 (Weak bisimulation) *A relation $\mathcal{R}$ is a weak bisimulation if it is symmetric and $S\ \mathcal{R}\ T$ implies If $S \overset{m}{\longrightarrow} S'$ then there exists a T' s.t. $T \overset{\hat{m}}{\Longrightarrow} T'$ and $S'\ \mathcal{R}\ T'$ where $\hat{\tau} = \epsilon$ and $\hat{m} = m$ for $m \neq \tau$. Two programs S and T are weak bisimilar, written $S \dot{\approx} T$, if $S\ \mathcal{R}\ T$ for some weak bisimulation $\mathcal{R}$.*

4 Equational Theories for Finite Plain LAL

In the rest of the paper we shall present sound and complete axiomatizations of strong and weak bisimulation for finite Plain LAL, i.e. Plain LAL without replication. The usual way of proving completeness is by means of a normal form to which all bisimilar agents can be converted. This is not immediately possible in LAL as sum and parallel composition operate on two different levels. To deal with this, we extend the syntax of programs with prefixing and sum:

$$S ::= \sum_{i \in I} m_i.S_i \mid \cdots \qquad \text{where } m ::= \tau \mid \alpha?\beta$$

Observe that we do not have spawn prefixes at this level as these are seen as τ-actions from the outside and can be modelled as such. Since the set of prefixes at system level is a subset of the set of labels we use the same letters as metavariables. With the extended syntax we get both sum and composition at system level and this enables us to define expansion theorems. We define the semantics of this new construct by extending the transition relation $\longrightarrow$ with the following rule:

$$\sum_{i \in I} m_i.S_i \overset{m_j}{\longrightarrow} S_j \text{ if } j \in I$$

and we define $\sum_{\emptyset} S = \bullet$.

4.1 Axiomatization of Strong Bisimulation

In this section we shall give a complete axiomatization of $\sim$ for Plain LAL agents of the extended syntax.

We shall need to decide whether a restricted name can become accessible from outside, such that it can be used to get a value. Intuitively, a name can become accessible if it is located at an unrestricted location. With this definition α can become accessible in $(\alpha)((\beta,\alpha)|(\alpha,\gamma))$ but not in $(\alpha)(\sigma)\,((\beta,\sigma)|(\sigma,\alpha)|(\alpha,\gamma))$. But in the last expression α can clearly become accessible through β and σ — we therefore need to be able to follow a sequence of names and to this end we define a notion of *reachability*:

Definition 6 (Reachability) *Let $S = (L)\prod(\alpha_i,\beta_i)$. We say α_j is reachable in S if either α_j is free or there exists a sequence $\gamma_1,\ldots,\gamma_n$ where for $1 \leq i < n$ we have for some k: $\gamma_i = \alpha_k$ and $\gamma_{i+1} = \beta_k$, further γ_1 is free and $\gamma_n = \alpha_j$. If α is not reachable in S we say that α is* unreachable *in S.*

Observe that if α is not a location of a value we consider it unreachable. Further observe that reachability of a name is a strictly syntactic predicate, thus it is always decidable.

The axioms are given in Table 2

We shall denote by $\mathcal{A}$ this set of rules and axioms, and we write $\mathcal{A} \vdash S = T$ if $S = T$ can be proven using equational reasoning over the rules of $\mathcal{A}$.

4.2 Soundness and Completeness

Theorem 5 (Soundness) *If $\mathcal{A} \vdash S = T$ then $S \sim T$*

Proof. We just need to show that the axioms hold if we replace $=$ by $\sim$. □

Completeness is established through the concept of head normal forms and the notion of *depth* of an agent. The depth of an agent is the maximal length of any transition sequence that the agent can perform (a precise definition can be found in [HK94].)

Definition 7 (Head normal form) *S is on* head normal form*, abbreviated hnf., if*

$$S = (L)\left(\sum m_i.S_i \;\middle|\; \prod\,(\alpha_j,\beta_j)\right)$$

where $L \subseteq \bigcup\{\beta_j\}$, $L \backslash \bigcup\{\beta_j\} \cap \mathsf{fn}(m_i) = \emptyset$ and for all k in the index set is α_k reachable in $(L)\prod(\alpha_j,\beta_j)$.

$C1 \quad S = T \quad \text{if } S \equiv T$

$C2 \quad \alpha?\beta.S = \alpha?\beta.T \quad \text{if } S = T$

$C3 \quad \tau.S = \tau.T \quad \text{if } S = T$

$C4 \quad (\alpha)S = (\alpha)T \quad \text{if } S = T$

$C5 \quad S_1|S_2 = T_1|T_2 \quad \text{if } S_1 = T_1 \text{ and } S_2 = T_2$

$C6 \quad \sum_I m_i.S_i = \sum_I n_i.T_i \text{ if } \forall j \in I : m_j.S_j = n_j.T_j$

$$S1 \quad \left(\alpha, \sum a_i.P_i\right) = \sum m_i.S_i \text{ where } \begin{cases} m_i = a_i, S_i = (\alpha, P_i) & \text{if } a_i = \tau, \beta?\gamma \\ m_i = \tau, S_i = (\alpha, P_i)|(\beta, Q) & \text{if } a_i = \beta!Q \end{cases}$$

$$S2 \quad \sum_I S_i = \sum_{I\backslash\{k\}} S_i \quad \text{if } S_j = S_k \text{ for some } j \in I\backslash\{k\}$$

$$S3 \quad (\alpha)\sum m_i.S_i = \sum m_i.(\alpha)S_i \quad \text{if } \alpha \notin \bigcup \mathsf{n}(m_i)$$

$$S4 \quad (\alpha)\sum m_i.S_i = (\alpha) \sum_{m_i \neq \alpha?\beta} m_i.S_i$$

$$E1 \quad S|T = \sum m_i.(S_i|T) + \sum n_j.(S|T_j)$$
$$\text{if } S = \sum m_i.S_i,\ T = \sum n_j.T_j \text{ and } \mathsf{bn}(m_i) \cap \mathsf{fn}(T) = \emptyset = \mathsf{bn}(n_j) \cap \mathsf{fn}(S)$$

$$E2 \quad (L)\left(\sum m_i.S_i \mid (\alpha,\beta)| \prod(\alpha_j,\beta_j)\right) = (L)\left(\sum_{m_i \neq \alpha?\gamma} m_i.\,(S_i|(\alpha,\beta)) + \sum_{m_i = \alpha?\gamma} \tau.S_i\{\beta/\gamma\} \;\Big|\; \prod(\alpha_j,\beta_j)\right)$$
$$\text{if } \alpha \text{ is unreachable in } (L)\prod(\alpha_j,\beta_j)$$

Table 2: The equational theory for strong bisimilarity

Lemma 6 *If S if a finite Plain* LAL *process then there exists a hnf. H of no greater depth s.t. $\mathcal{A} \vdash S = H$.*

Proof. Induction in the structure of S. □

We can now prove completeness:

Theorem 7 (Completeness) *Let S and T be finite Plain* LAL *terms. If $S \sim T$ then $\mathcal{A} \vdash S = T$*

Proof. We have $S \sim T$. The proof is by induction in the maximal depth of S and T. □

4.3 Axiomatization of Weak Bisimulation

It is well-known that weak bisimulation is not a congruence, not even if we close it under substitutions, as it is not preserved by summation. For that reason we shall introduce another bisimulation similar to Milner's observation equality.

Definition 8 *Two agents S and T are observation equivalent, written $S \asymp T$ if*

- *Whenever $S \xrightarrow{m} S'$ then there exists a T' s.t. $T \overset{m}{\Longrightarrow} T'$ and $S' \dot{\approx} T'$.*

and vice versa.

Observe that if the first move is a τ transition it cannot be matched by an ε-transition. However, after the first move we only require weak bisimulation to hold. It can be shown that $\asymp$ is the induced congruence of $\dot{\approx}$.

Observation equivalence is related to weak bisimulation as stated by the following proposition:

Proposition 8 *$S \dot{\approx} T$ iff $S \asymp T$, $S \asymp \tau.T$ or $\tau.S \asymp T$.*

Proof. As in [Mil89] □

Normally, when defining normal forms for completeness proofs one eliminates parallel composition. In the previous section we saw that we could allow a limited use of parallel composition and still be able to prove completeness in the case of strong bisimulation. We have not been able to accomplish this for weak bisimulation. To get around this we extend the prefixes of agents to incorporate all labels of the labelled transition system; this does not require any changes in the semantics.

The axiom system $\mathcal{AA}$ consists of the axioms $C1$, $C3 - C6$ and $S1 - S2$ from the axiom system $\mathcal{A}$ and the following six:

$C2$ $\quad \alpha?\gamma.S = \alpha?\gamma.T$ if $S\{\beta/\gamma\} = T\{\beta/\gamma\}$ for all $\beta \in \mathsf{fn}(S|T) \cup \{\gamma\}$

$E1$ $\quad (\alpha,\beta) = \beta@\alpha.\mathbf{0}$

$E2$ $$S|T = \sum m_i.(S_i|T) + \sum n_j.(S|T_j) + \sum_{\substack{m_i = \alpha?\gamma \\ n_j = \beta@\alpha}} \tau.(S_i\{\beta/\gamma\}|T_j) + \sum_{\substack{m_i = \beta@\alpha \\ n_j = \alpha?\gamma}} \tau.(S_i|T_j\{\beta/\gamma\})$$
if $S = \sum m_i.S_i,\ T = \sum n_j.T_j$ and $\mathsf{bn}(m_i) \cap \mathsf{fn}(T) = \emptyset = \mathsf{bn}(n_j) \cap \mathsf{fn}(S)$

$S3$ $$(\beta)\sum m_i.S_i = \sum_{\beta \notin \mathsf{fn}(m_i)} m_i.(\beta)S_i \ + \sum_{\substack{m_i = \beta@\alpha_i \\ \beta \neq \alpha_i}} (\beta)@\alpha_i.S_i$$

$T1$ $\quad S + \tau.S = S$ if S is a summation

$T2$ $\quad m.\tau.S = m.S$

The axioms $T1$ and $T2$ are similar to the well-known τ-laws of CCS; the third CCS τ-law is concerned with τ-transitions after a visible action and is thus not relevant in our case. The axiom $C2$ is necessary because in the extended syntax we no longer have preservation of bisimulation under substitution, as we now end up with a synchronous calculus. Alternatively we could have limited the set of processes to processes which are bisimilar to processes of Plain LAL.

Theorem 9 *If* $\mathcal{AA} \vdash S = T$ *then* $S \asymp T$

Proof. The axioms hold then replacing $=$ by $\asymp$, and the rules preserve soundness. $\square$

To prove completeness we introduce standard forms:

Definition 9 *S is in standard form if*

$$S = \sum m_i.S_i$$

and the S_i's are in standard form.

Lemma 10 *For every finite agent S there there exists a standard form T of no greater depth s.t. $\mathcal{AA} \vdash S = T$.*

Proof. The proof goes by induction in the structure of S. □

As is usual, we need to refine the notion of standard form to full standard form:

Definition 10 *A standard form S is a full standard form if $S \stackrel{m}{\Longrightarrow} S'$ implies $S \xrightarrow{m} S'$.*

Lemma 11 *For every finite agent S there there exists a full standard form T of no greater depth s.t. $\mathcal{AA} \vdash S = T$.*

We now get

Theorem 12 (Completeness) *If $S \asymp T$ then $\mathcal{AA} \vdash S = T$.*

Proof. By Lemma 11 we can assume that S and T are full standard forms. The proof proceeds by induction in the depths of S and T. □

We have now shown how to axiomatize $\asymp$, though it was $\dot{\approx}$ we really wanted to axiomatize. However, by Proposition 8 we have $S \dot{\approx} T$ iff either $\mathcal{AA} \vdash S = T$, $\mathcal{AA} \vdash S = \tau.T$ or $\mathcal{AA} \vdash \tau.S = T$.

It is obvious that two processes can effectively be put in standard form. Furthermore by inspection of the proof of Theorem 7 and Theorem 12 we see that it can effectively be checked whether two processes in standard form are bisimilar and weak bisimilar respectively. It now follows that $S \sim T$ is decidable, furthermore $S \asymp T$ is decidable and hence $S \dot{\approx} T$ is decidable.

Observe that we need to extend the calculus not because it is asynchronous but because it is two-level. Compared to the axiomatization of the π-calculus in [PS93] we see that we end up with almost the same axioms. The main difference being that we have a simpler rule for input prexing in the strong case.

5 Conclusions and Related Work

In this paper we have put forward an asynchronous process calculus, Plain LAL, and defined the notions of early, late and open bisimulation equivalence within the setting of the operational semantics provided. It turns out that these notions of bisimulation coincide here. We give a sound and complete axiomatization of the common notion of bisimulation equivalence for finite Plain LAL agents.

Plain LAL looks somewhat different from the calcui of [HT91] and [Bou92] but it is in fact quite easy to come up with compositional encodings of these calculi in Plain LAL. Although we have not persued this subject very further, this leeds us to belive that our results will indeed also hold for asynchronous versions of the π-calculus.

An interesting question is: How much can we extend Plain LAL before the three definitions of bisimulation differ? For instance, if we introduce the matching operator of the π-calculus, the two agents $(\alpha, \tau.\beta! + \tau)$ and $(\alpha, \tau.\beta! + \tau + \tau.[\beta = \gamma]\beta!)$ are late bisimilar but not open.

The ideas from Plain LAL can be extended to a higher-order calculus which allows the passing of processes. This calculus, called LAL, is similar to CHOCS ([Tho90]) and has been studied in [HK94]. It turns out that Corollary 3 also holds for LAL. Further, there is an adequate translation of LAL into Plain LAL – thus, the equational theory for Plain LAL can be used in conjunction with the translation to provide equational reasoning for LAL agents.

One should also note an important difference between our calculus and the fork calculus studied by [Hav94] which seems to indicate that an implicit parallel operator works better in the setting of asynchrony. For in the setting of the fork calculus the formulation of the bisimulation congruence becomes quite difficult, whereas the bisimulation congruence of Plain LALis entirely straightforward, being the bisimulation equivalence itself.

References

[BMT92] D. Berry, R. Milner, and D.N. Turner. A semantics for ml concurrency primitives. *Conference Record of the 19th Annual ACM Symposium on Principles of Programming Languages*, January 1992.

[BN92] Michele Boreale and Rocco De Nicola. Testing equivalences for mobile processes. In W.R. Cleaveland, editor, *CONCUR '92*, number 630 in Springer LNCS, pages 2–16. Springer-Verlag, 1992.

[Bou92] Gérard Boudol. Asynchrony and the π-calculus. Technical report, INRIA Sophia-Antipolis, 1992.

[GB82] David Gelernter and Arthur J. Bernstein. Distributed communication via global buffer. In *Proceedings of ACM Symposium on Principles of Distributed Computing*, pages 10–18, August 1982.

[Hav94] Klaus Havelund. *The Fork Calculus.* PhD thesis, Department of Computer Science, University of Copenhagen, January 1994.

[HK94] Martin Hansen and Josva Kleist. Process calculi with asynchronous communication. Master's thesis, Aalborg University, 1994.

[HT91] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In *ECOOP 91*, volume 512 of *Lecture Notes in Computer Science*, pages 133–147. Springer-Verlag, 1991.

[HY93] Kohei Honda and Nobuko Yoshida. On reduction-based process semantics. In *Proceedings of FST & TCS*, volume 761 of *Lecture Notes in Computer Science*, pages 373–387. Springer, 1993.

[Ing94] Anna Ingólfsdóttir. Late and early semantics coincide for testing. *Theoretical Computer Science*, 1994. To appear.

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

[MPW89] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes — part I and II. Technical Report ECS-LFCS-89-85 and -86, Department of Computer Science, University of Edinburgh, 1989. Also published in *Information and Computation*, 100:1-77,1992.

[Ode95] Martin Odersky. Applying π: Towards a basis for concurrent imperative programming. In U.S. Reddy, editor, *Proceedings of Second ACM SIGPLAN Workshop on State in Programming Languages (SIPL'95)*, pages 95–108. Dept. of Computer Science, Univ. of Illinois, January 1995. Technical Report no. UIUCDCS-R-95-1900.

[PS93] Joachim Parrow and Davide Sangiorgi. Algebraic theories for name-passing calculi. *Information and Computation*, 1993. To appear. Also available as Report ECS-LFCS-93-262, Department of Computer Science, University of Edinburgh, 1993.

[San93] Davide Sangiorgi. A theory of bisimulation for the π-calculus. In E. Best, editor, *Proceedings of CONCUR 93*, volume 715 of *Lecture Notes in Computer Science*, pages 127–142. Springer-Verlag, 1993.

[Tho90] Bent Thomsen. *Calculi for Higher-Order Communicating Systems.* PhD thesis, Imperial College, University of London, 1990.