Trios in Concert

Joachim Parrow
Royal Inst. of Technology, Dep. Teleinformatics
Electrum 204, S-16440 Kista, Sweden.
email joachim@sics.se

July 22, 1996

Abstract

A *trio* is a term $\alpha.\beta.\gamma.\mathbf{0}$ in the polyadic π-calculus. We show that restricted parallel composition of possibly replicated trios, where at most one contains a bound output prefix, is enough to obtain the expressive power of the full summation-free π-calculus up to weak bisimulation equivalence. Therefore that fragment of the calculus is undecidable.

1 Introduction

The π-calculus [MPW92, Mil91] is a basic calculus for describing reactive processes, i.e., processes that continually interact with an environment. Although there is yet no definite measure on its expressiveness there is strong evidence that its primitives are enough for a wide variety of purposes. Within the π-calculus it is possible to naturally encode the functional paradigms of the λ-calculus [Mil92] and of object oriented formalisms [Wal95]. The ability to directly represent mobility, in the sense of processes that reconfigure their interconnection structure when they execute, makes it easy to model systems where processes move between different locations and where resources are allocated dynamically [OP92, Ora94]. It can also naturally encode higher-order communication where processes are transmitted in communications [San93] and in this way lays the foundation for many programming languages.

The expressive power of the π-calculus over other process algebras such as CCS [Mil89] or ACP [BW90] has been debated. All such algebras have a construct $P|Q$ which represents P and Q executing in parallel. One feature which distinguishes the π-calculus from other formalisms is the ability of a process to invent a new port name and send that name to other processes. This is achieved with the restriction operator, $(\nu x)P$ (akin to CCS restriction, $P\backslash x$) together with the output prefix $\overline{a}x$; when these are combined into $(\nu x)\overline{a}x\,.\,P$ we have a process emitting a newly invented name. Since that process may itself be under the scope of an iterative construct such as replication, written "!", we can formulate processes such as $!\,(\nu x)\overline{a}x\,.\,P$ which provide an unlimited supply of new names.

The π-calculus is an algebra in the sense that terms are built as arbitrary combinations of such constructs. For example, replication can operate on terms of any size and new names can be generated at several places, so terms can be quite complex. It is then natural to ask how much of this complexity is necessary in order to attain the full expressive power. In this paper I shall present a fragment of the calculus, in the form of a subset of its terms, such that any π-calculus term is weakly equivalent to a term in the fragment. This throws light on exactly what it is that makes the π-calculus so powerful. Another consequence is that all interesting decision problems for that fragment are undecidable.

The fragment is formed as follows. Prefix operators are limited to occur in *trios*, terms of the form $\alpha\,.\,\beta\,.\,\gamma\,.\,\mathbf{0}$. A trio is thus capable of at most three interactions. Replication only operates on such trios. The possibly replicated trios are combined through the parallel operator, together with a term $!\,(\nu x)\overline{a}x$ which invents new names. The main idea is that in order to emulate an arbitrary π-calculus term P it is enough to have one trio for each subterm of P, controlling the activities of that subterm. These trios work in concert to ensure that the right subterms execute at the right time.

There appears to be little work directly aimed at finding such fully expressible fragments of process algebras. Related efforts within the π-calculus are Sangiorgi's result that guarded replication can replace all instances of unguarded replication while preserving strong equivalence [San94] and Honda and Tokoro's work on asynchronous communication [HT92], although neither of these attempt to reduce terms to the simplest possible parallel components. Such a reduction is easier to accomplish with a synchronous parallel operator, allowing multiway rendezvous, because then one component can

interact with several other simultaneously. Examples of results in this direction are my investigation of a synchronous process algebra [Par90] where three basic terms combined in parallel generate all finite-control terms, and Gonthier's normal forms for MEIJE [Gon85]. Vaandrager [Vaa93] gives a general account of expressiveness in process algebras, where much effort has focussed on operators rather than terms following the pioneering work of de Simone [dS85]. In other models of computation this kind of result is more common, for example within the λ-calculus a small set of combinators is enough to express any closed λ-term [Bar84].

The following three sections contain our notational conventions, an explanation of the translation into trios, and the proof of the main result. This amounts to a rather technical exercise; a concluding more general section points out the main implications and variants of the construction.

2 Notation

To appreciate the following sections the reader should have some previous experience with the π-calculus. This section points out the particular notation and conventions used here.

We will work with the polyadic π-calculus without summation and without matching, and with replication rather than recursion. So assume a countable set of names ranged over by $a, b, \ldots, x, y, \ldots$ and let $\widetilde{u}$, $\widetilde{v}$,... range over sequences of names. The syntax of π-calculus terms, ranged over by $P, Q, \ldots$ is given by

$$\begin{array}{lll} P ::= & \mathbf{0} & \text{(inaction)} \\ & \tau\,.\,Q & \text{(silent prefix)} \\ & \overline{a}\widetilde{v}\,.\,Q & \text{(output prefix)} \\ & a(\widetilde{v})\,.\,Q & \text{(input prefix)} \\ & (\nu x)Q & \text{(restriction)} \\ & Q \mid R & \text{(parallel composition)} \\ & !\,Q & \text{(replication)} \end{array}$$

In the input prefix the members of $\widetilde{v}$ must be pairwise distinct. Restriction and input bind names, and the notions of free and bound names are standard as is the notion of substitution of the name x for the free occurrences of y in P, written $P\{x/y\}$, generalized to substitutions of sequences of equal length

$$\alpha.P \xrightarrow{\alpha} P \qquad \frac{!P \mid P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'}$$

$$\frac{P \xrightarrow{\overline{a}(\nu\widetilde{x})\widetilde{z}} P', \quad Q \xrightarrow{a(\widetilde{v})} Q', \quad |\widetilde{z}| = |\widetilde{v}|}{P|Q \xrightarrow{\tau} (\nu\widetilde{x})(P' \mid Q'\{\widetilde{z}/\widetilde{v}\})} \qquad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

$$\frac{P \xrightarrow{\overline{a}(\nu\widetilde{x})\widetilde{z}} P', \quad y \in \widetilde{z}}{(\nu y)P \xrightarrow{\overline{a}(\nu y\widetilde{x})\widetilde{z}} P'} \qquad \frac{P \xrightarrow{\alpha} P', \quad y \notin \alpha}{(\nu y)P \xrightarrow{\alpha} (\nu y)P'}$$

Table 1: Transition relations of agents. We assume that all involved bound names are distinct. Alpha-variants of agents are considered identical and so have the same transitions. The rules for parallel composition have symmetric forms.

$P\{\widetilde{w}/\widetilde{u}\}$ (where the members of $\widetilde{u}$ are pairwise distinct). We will also use substitutions on prefixes and names with the obvious interpretation. The empty sequence is written ϵ and can be omitted in the prefix forms (so, e.g., $\overline{a}\epsilon$ can be written $\overline{a}$) and the length of $\widetilde{u}$ is $|\widetilde{u}|$. The abbreviation $(\nu z_1 \ldots z_n)P$ means $(\nu z_1)\ldots(\nu z_1)P$, in particular $(\nu\epsilon)P$ means just P.

We will not distinguish between alpha-equivalent terms, and therefore whenever convenient assume that in each term every bound name is distinct from all other names in the term.

The semantics of terms is given in a family of transition relations $\xrightarrow{\alpha}$ where α is an action, i.e, either τ, the silent action, or $\overline{a}(\nu\widetilde{x})\widetilde{v}$, where all elements in $\widetilde{x}$ are also in $\widetilde{v}$, an output action, or $a(\widetilde{v})$, an input action, where elements of $\widetilde{v}$ are pairwise distinct. The *subject* of the input and output action above is a, and the *object* is the rest of the action. Note that the object of the output action contains not only a sequence of names $(\widetilde{v})$ but also a binding of a subset of them $(\nu\widetilde{x})$. If $\widetilde{x} = \epsilon$ we omit ν and the brackets and write the output action simply $\overline{a}\widetilde{v}$.

The transition relations are defined in the way which is now standard in the literature on the π-calculus, see Table 1. We use the "late" version of the semantics, where the rule for input prefix does not instantiate the bound names. There is a scope opening rule for restriction which adds a binding in the object of an output action. There is also a rule for inferring communica-

tion between parallel components which outputs respectively inputs objects of equal length on the same subject. This entails a substitution of the output object for the input object and adds a restriction for each name bound in the output object. The rule for replication simply says that $!\,P$ has the same transitions as $(!\,P) \mid P$.

We will use the *open* bisimulation equivalences originally proposed by Sangiorgi [San96]. A *strong* open bisimulation $\mathcal{S}$ is a binary relation on agents such that if $P\mathcal{S}Q$ then for every transition $P \xrightarrow{\alpha} P'$ (where the names bound in α do not occur free in P or Q) there is a simulating transition $Q \xrightarrow{\alpha} Q'$ with $P'\mathcal{S}Q'$ and vice versa; moreover it must also hold that $P\{x/y\}\mathcal{S}Q\{x/y\}$ for all x, y, i.e., the relation is closed under arbitrary substitution[1]. Two agents P and Q are *strongly equivalent*, written $P \sim Q$, if $P\mathcal{S}Q$ for some strong open bisimulation.

As usual we let $\stackrel{\widehat{\tau}}{\Longrightarrow}$, or sometimes $\Longrightarrow$, mean $(\xrightarrow{\tau})^n$ for some $n \geq 0$ and, for $\alpha \neq \tau$, we let $\stackrel{\widehat{\alpha}}{\Longrightarrow}$ or $\stackrel{\alpha}{\Longrightarrow}$ mean $\Longrightarrow \xrightarrow{\alpha} \Longrightarrow$. The *weak* (open) equivalence, $\approx$, is defined in terms of weak open bisimulations where the simulating transition for $P \xrightarrow{\alpha} P'$ is $Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q'$. Both $\sim$ and $\approx$ are congruences in this version of the π-calculus, and they are strictly finer than the "early" and "late" congruences.

3 Trios and Concerts

A *trio* is a π-calculus term consisting of exactly three nested prefixes, i.e., of the form $\alpha\,.\,\beta\,.\,\gamma\,.\,\mathbf{0}$. We will in trios also admit a derived form of output prefix $\overline{a}(\nu x)x\,.\,P$, defined to mean $(\nu x)\overline{a}x\,.\,P$, outputting a single bound name along a. A trio containing such a derived prefix is called an *inventive* trio (it invents a new name x and makes that available to its environment). A *degenerate* trio contains less than three prefixes. A degenerate trio is of course weakly equivalent to the proper trio formed by filling the missing prefixes with τ, so we will include the degenerate trios among the trios. A trailing $\mathbf{0}$ will be omitted. Therefore, e.g., α is also a (degenerate) trio, corresponding to the proper trio $\alpha\,.\,\tau\,.\,\tau\,.\,\mathbf{0}$.

[1] Sangiorgi's original formulation was slightly different but the present definition yields the same equivalence and is easier in the context of this paper.

A *concert* of trios is a restriction of a parallel composition of possibly replicated trios, i.e., an agent of type

$$(\nu\tilde{z})((!)T_1 \mid \cdots \mid (!)T_n)$$

where each T_i is a trio, and at most one of the T_i is inventive. The main result in this paper is that for each agent there is a weakly equivalent concert. The idea is that the concert for P will have one trio for each subterm of P. This trio is responsible for "enacting" that subterm when appropriate. Since this may involve enacting further subterms — for example, enacting $P_1|P_2$ entails enacting P_1 and P_2 — the trio may call on other trios to do this. Furthermore, the only inventive trio will be a general "name provider" which sends new names to other trios at need.

To make this work we must assume a unique set of names through which the trios can call each other. We do this as follows. To each term P we associate a new unique *trigger* name z_P. The intuition is that transmitting something along z_P will invoke the trio responsible for the execution of P. We further introduce one new name z_ν. It is along this name that the name provider will send its new names to other trios. These trigger names are all distinct. Formally this means that we extend the calculus by introducing these names; at the end of this section we shall see that this extension, although convenient, is not strictly necesseray for the main result.

Definition 1 *Let P be an agent where we assume that all bound names are unique, and let $\tilde{u}$ be a sequence of pairwise distinct names. The* $\tilde{u}$-breakdown *of P, written $\mathcal{B}_{\tilde{u}}(P)$ is an agent defined inductively as follows.*

P	$\mathcal{B}_{\tilde{u}}(P)$	
$\mathbf{0}$	$z_{\mathbf{0}}(\tilde{u})$	
$\tau\,.\,Q$	$z_P(\tilde{u})\,.\,\tau\,.\,\overline{z_Q}\tilde{u}$	$\mid \mathcal{B}_{\tilde{u}}(Q)$
$\overline{a}\tilde{v}\,.\,Q$	$z_P(\tilde{u})\,.\,\overline{a}\tilde{v}\,.\,\overline{z_Q}\tilde{u}$	$\mid \mathcal{B}_{\tilde{u}}(Q)$
$a(\tilde{v})\,.\,Q$	$z_P(\tilde{u})\,.\,a(\tilde{v})\,.\,\overline{z_Q}\tilde{u}\tilde{v}$	$\mid \mathcal{B}_{\tilde{u}\tilde{v}}(Q)$
$(\nu x)Q$	$z_P(\tilde{u})\,.\,z_\nu(x)\,.\,\overline{z_Q}\tilde{u}x$	$\mid \mathcal{B}_{\tilde{u}x}(Q)$
$Q \mid R$	$z_P(\tilde{u})\,.\,\overline{z_Q}\tilde{u}\,.\,\overline{z_R}\tilde{u}$	$\mid \mathcal{B}_{\tilde{u}}(Q) \mid \mathcal{B}_{\tilde{u}}(R)$
$!Q$	$!\,z_P(\tilde{u})\,.\,\overline{z_P}\tilde{u}\,.\,\overline{z_Q}\tilde{u}$	$\mid\, !\mathcal{B}_{\tilde{u}}(Q)$

Notice that $\mathcal{B}_{\tilde{u}}$ binds the names $\tilde{u}$. The breakdown of P consists of a (possibly replicated) leading trio, in parallel with the breakdown of the subterms of

P. The leading trio is called a *prefix* trio if P is a prefix form, otherwise it is called a *control* trio. So, using $!(P|Q) \sim !P|!Q$ and $!!P \sim !P$, we can push all replication inwards in $\mathcal{B}_{\tilde{u}}(P)$ until it becomes a parallel composition of possibly replicated trios, one for each subterm of P. There will be one prefix trio for each prefix in P, and one control trio for each other subterm in P. Each of these has a unique trigger.

The object $\tilde{u}$ corresponds to the name instantiations which are needed when executing a subterm. These names are bound outside the subterm, in other words they are placeholders for something which will be determined outside the subterm. The intention is that $\mathcal{B}_{\tilde{u}}(P)$ in parallel with a trigger $\overline{z_P}\tilde{w}$ will enact $P\{\tilde{w}/\tilde{u}\}$. If we start with $\mathcal{B}_\epsilon(P)$ then each subterm Q in P will be broken down to $\mathcal{B}_{\tilde{u}}(Q)$, where $\tilde{u}$ is the sequence of bindings outside Q whose scope extend into Q. It may appear strange that $\tilde{u}$ are bound in $\mathcal{B}_{\tilde{u}}(Q)$ whereas they are free in Q. But in the definition of $\mathcal{B}$ we are interested in Q *as a subterm of* P (it may be clearer to think of it as $\mathcal{B}_{P,\tilde{u}}(Q)$, even though the definition turns out to be independent of P) and although $\tilde{u}$ are free in Q they are bound in P. Of course, using alpha-conversion they can be renamed so the definition of $\mathcal{B}_{\tilde{u}}(Q)$ really only depends on the length of $\tilde{u}$. But note that a in the clauses for output and input (and also $\tilde{v}$ in the clause for output) may be among $\tilde{u}$.

The leading trio corresponding to $\mathcal{B}_{\tilde{u}}(P)$ begins by awaiting, on its trigger z_P, the reception of the names which shall instantiate the bound names $\tilde{u}$. When those have been received the trio takes different actions depending on the form of P. If P is a prefix $\alpha\,.\,Q$ the trio will enact α and then activate Q through its trigger, forwarding $\tilde{u}$. If α is an input prefix then the sequence of bound names will grow (bound names are unique so $\tilde{u}$ and $\tilde{v}$ will be disjoint), and therefore the new names received in the input ($\tilde{v}$) will be appended to the names transferred into Q. For $\alpha = \tau$ we could actually have simplified the definition to $\mathcal{B}_{\tilde{u}}(\tau\,.\,Q) = \mathcal{B}_{\tilde{u}}(Q)$ (since $Q \approx \tau\,.\,Q$) at the expense of the symmetric treatment of the prefixes.

If P is a restriction $(\nu x)Q$, then it proceeds to collect along z_ν a new name, which it appends to the names transferred into Q. The intention is that $\mathcal{B}_{\tilde{u}}(P)$ is executed in parallel with a name provider sending new names along z_ν.

If P is a parallel composition then its control trio simply activates the factors of P. For this to work it is important that each trio begins with a reception along its trigger. If for example the breakdown of $\mathbf{0}$ would be $\mathbf{0}$

(rather than $z_{\mathbf{0}}(\tilde{u})$) then $\mathcal{B}_{\tilde{u}}(\mathbf{0} \mid R)$ would never reach the point where R is activated.

Finally, if P is a replication $!Q$, its replicated control trio does two things: first it activates a new incarnation of itself and then it activates (an incarnation of) Q. Because of the replications this means that arbitrarily many copies of Q can be started. It would not be enough to define $\mathcal{B}_{\tilde{u}}(!Q) = !\mathcal{B}_{\tilde{u}}(Q)$; in that case one activation of $\mathcal{B}_{\tilde{u}}(!Q)$ would only result in one activation of Q.

It is worth explaining a possible controversy here when P has a subterm occurring in several positions, as in $(\overline{a}\,.\,\overline{b}) \mid \overline{b}$. In the breakdown of that term there will be two trios triggered by $z_{\overline{b}}$, one for each occurrence of $\overline{b}$. As it turns out this is harmless, since whenever someone signals on $z_{\overline{b}}$, the execution of $\overline{b}$ will be started in one of the trios — it does not matter which one (see the proof of Lemma 6 in the next section)! Note that our assumption about unique bound names forbids agents like $a(x)\,.\,\overline{x}y \mid a(y)\,.\,\overline{x}y$ where a subterm $\overline{x}y$ occurs in different places with different bound names (in that case it would really matter which trio was activated).

The translation of an agent into a concert now goes as follows.

Definition 2 *Let the* name inventor N *be defined by*

$$N = !\overline{z_\nu}(\nu x)x$$

and let $Z(P)$, the triggers of P, be defined by

$$Z(P) = \{z_\nu\} \cup \{z_Q : Q \textit{ is a subterm of } P\}.$$

The translation $\mathcal{T}(P)$ *of an agent P is defined to be*

$$\mathcal{T}(P) = (\nu Z(P))(\mathcal{B}_\epsilon(P) \mid \overline{z_P} \mid N)$$

Restriction is commutative so restricting on a set of names is a well defined operation, up to strong equivalence. Note that $\mathcal{T}(P)$ is equivalent (using the laws $!(P|Q) \sim !P|!Q$ and $!!P \sim !P$) to a restricted parallel composition of possibly replicated trios where only N is inventive, i.e.,$\mathcal{T}(P)$ is equivalent to a concert.

Theorem 3 $P \approx \mathcal{T}(P)$.

The idea behind the proof is that $(\nu Z(P))(\mathcal{B}_{\widetilde{u}}(P) \mid \overline{z_P}\widetilde{w} \mid N)$ is bisimilar to $P\{\widetilde{w}/\widetilde{u}\}$ for all $\widetilde{u}, \widetilde{w}$ of equal length. The proof is simply to establish a bisimulation relating these agents, by showing how their transitions correspond to each other. The details are contained in the next section.

The definition of $\mathcal{T}(P)$ uses the trigger names only in bound positions. Since the set of names is infinite and each $\mathcal{T}(P)$ only contains a finite set of names, there is an alpha-variant of $\mathcal{T}(P)$ which does not use the trigger names. So the extension of the caclulus to include these names is not strictly necessary, although it simplifies the proof.

A variant of the same result is to redefine $\mathcal{B}_{\widetilde{u}}$ so that *all* trios in $\mathcal{B}_{\widetilde{u}}(P)$ are replicated. Since a trio only starts executing when signalled to do so it is harmless to replicate it. The concert will then be more uniform, consisting of replicated trios plus only one non-replicated trio (the trigger $\overline{z_P}$ in $\mathcal{T}(P)$).

4 Proof

To prove the theorem we establish a weak (open) bisimulation including $(P, \mathcal{T}(P))$. To do this, we begin by defining the agents *correlated* to $P\{\widetilde{w}/\widetilde{u}\}$ in this bisimulation. These are essentially the agents obtained by starting $\mathcal{B}_{\widetilde{u}}(P)$ by sending $\widetilde{w}$ along the trigger for P.

Definition 4 *Assume $\widetilde{w}$ and $\widetilde{u}$ have equal length, and $\widetilde{u}$ are pairwise distinct. The set of agents $\mathcal{C}^{\widetilde{w}}_{\widetilde{u}}(P)$ is defined to be*

$$\mathcal{C}^{\widetilde{w}}_{\widetilde{u}}(P) = \{\overline{z_P}\widetilde{w} \mid \mathcal{B}_{\widetilde{u}}(P)\} \quad \cup \quad \mathcal{D}^{\widetilde{w}}_{\widetilde{u}}(P)$$

where $\mathcal{D}^{\widetilde{w}}_{\widetilde{u}}(P)$ *is defined inductively as follows:*

P	$\mathcal{D}^{\widetilde{w}}_{\widetilde{u}}(P)$
$\mathbf{0}$	$\{\mathbf{0}\}$
$\tau\,.\,Q$	$\{\tau\,.\,\overline{z_Q}\widetilde{w} \mid \mathcal{B}_{\widetilde{u}}(Q)\}$
$\overline{a}\widetilde{v}\,.\,Q$	$\{\overline{a}\widetilde{v}\{\widetilde{w}/\widetilde{u}\}\,.\,\overline{z_Q}\widetilde{w} \mid \mathcal{B}_{\widetilde{u}}(Q)\}$
$a(\widetilde{v})\,.\,Q$	$\{a(\widetilde{v})\{\widetilde{w}/\widetilde{u}\}\,.\,\overline{z_Q}\widetilde{w}\widetilde{v} \mid \mathcal{B}_{\widetilde{uv}}(Q)\}$
$(\nu x)Q$	$\{z_\nu(x)\,.\,\overline{z_Q}\widetilde{w}x \mid \mathcal{B}_{\widetilde{ux}}(Q)\}$
	$\cup\{(\nu x')C_Q : C_Q \in \mathcal{C}^{\widetilde{w}x'}_{\widetilde{ux}}(Q)\}$ $\quad (x'$ fresh$)$
$Q \mid R$	$\{\overline{z_Q}\widetilde{w}\,.\,\overline{z_R}\widetilde{w} \mid \mathcal{B}_{\widetilde{u}}(Q) \mid \mathcal{B}_{\widetilde{u}}(R)\}$
	$\cup\{(C_Q \mid C_R) : C_Q \in \mathcal{C}^{\widetilde{w}}_{\widetilde{u}}(Q), C_R \in \mathcal{C}^{\widetilde{w}}_{\widetilde{u}}(R)\}$
$!Q$	$\{(!\,z_P(\widetilde{u})\,.\,\overline{z_P}\widetilde{u}\,.\,\overline{z_Q}\widetilde{u} \mid \overline{z_P}\widetilde{w}\,.\,\overline{z_Q}\widetilde{w} \mid (\overline{z_Q}\widetilde{w} \mid)^n$
	$!\mathcal{B}_{\widetilde{u}}(Q) \mid C_1 \mid \cdots \mid C_m) : n, m \geq 0, C_i \in \mathcal{C}^{\widetilde{w}}_{\widetilde{u}}(Q)\}$

Now define the relation $\mathcal{S}$ by

$$\mathcal{S} = \{(P\{\widetilde{w}/\widetilde{u}\},\, R) : \begin{array}{l} R \in \{(\nu\widetilde{z})(N \mid C_P) : C_P \in \mathcal{C}^{\widetilde{w}}_{\widetilde{u}}(P)\}, \\ |\widetilde{w}| = |\widetilde{u}|,\ \widetilde{z} = Z(P)\} \end{array}$$

It is then clear that $(P, \mathcal{T}(P))$ is in $\mathcal{S}$. The significant properties are the following:

Lemma 5 *$\mathcal{S}$ is closed under substitution of names (up to strong equivalence).*

Proof: A straightforward and tedious examination of the definitions. Consider a pair $(P\{\widetilde{w}/\widetilde{u}\},\, R)$ where $R = (\nu\widetilde{z})(N \,|\, C_P)$ and $C_P \in \mathcal{C}^{\widetilde{w}}_{\widetilde{u}}(P)$. For a substitution $\{x/y\}$ we can assume without loss of generality that $x, y \notin \widetilde{z}$ (otherwise we have to alpha-convert R to satisfy this). So $R\{x/y\}$ is $(\nu\widetilde{z})(N \,|\, C_P\{x/y\})$. The important fact is that if $C_P \in \mathcal{C}^{\widetilde{w}}_{\widetilde{u}}(P)$ then $C_P\{x/y\} \in \mathcal{C}^{\widetilde{w}'}_{\widetilde{u}'}(P)$, where $\{\widetilde{w}'/\widetilde{u}'\} = \{\widetilde{w}/\widetilde{u}\}\{x/y\}$. To be precise this holds only up to a strong version of strong equivalence, which requires the inference of actions to have equal lengths (i.e., two agents are related if they have related derivatives inferred through proofs of equal length). However, that is enough for the purpose of the proofs below. □

Lemma 6 *Assume $P\{\widetilde{w}/\widetilde{u}\}\mathcal{S}R$. If $P\{\widetilde{w}/\widetilde{u}\} \xrightarrow{\alpha} P'$, then $R \Longrightarrow \xrightarrow{\alpha} R'$ and for some $\widetilde{x}$, $P' \sim (\nu\widetilde{x})P''$ and $R' \sim (\nu\widetilde{x})R''$ and $P''\mathcal{S}R''$.*

PROOF: The proof is an induction over the length of the inference of the transition in the antecedent. It consists of a rather long case analysis and only a few of the cases are shown here.

Consider the case where the antecedent is inferred through the rule for input prefix. Then $\alpha = a\{\widetilde{w}/\widetilde{u}\}(\widetilde{v})$ and $P = a(\widetilde{v}).Q$, where we without loss of generality assume that $\widetilde{v}$ has no element in common with $\widetilde{w}$ or $\widetilde{u}$ (otherwise we have to first alpha-convert P). Therefore $P' = Q\{\widetilde{w}/\widetilde{u}\}$. Now consider R. There are two possibilities, because $C^{\widetilde{w}}_{\widetilde{u}}(P)$ has two elements: $C_1 = \overline{z_P}\widetilde{w} \mid \mathcal{B}_{\widetilde{u}}(P)$ and $C_2 = a\{\widetilde{w}/\widetilde{u}\}(\widetilde{v}).\overline{z_Q}\widetilde{w}\widetilde{v} \mid \mathcal{B}_{\widetilde{uw}}(Q)$ (the latter comes from $\mathcal{D}^{\widetilde{w}}_{\widetilde{u}}(P)$ which here is a singleton). Put $R_i = (\nu\widetilde{z})(N \mid C_i)$. Clearly $R_1 \xrightarrow{\tau} R_2$. Also $R_2 \xrightarrow{\alpha} R'$ where $R' = (\nu\widetilde{z})(N \mid C')$ where $C' = \overline{z_Q}\widetilde{w}\widetilde{v} \,|\, \mathcal{B}_{\widetilde{uw}}(Q)$. So $C' \in \mathcal{C}^{\widetilde{w}\widetilde{v}}_{\widetilde{uv}}(Q)$. So R' is related in $\mathcal{S}$ to $Q\{\widetilde{w}\widetilde{v}/\widetilde{u}\widetilde{v}\} = Q\{\widetilde{w}/\widetilde{u}\}$, fulfilling the lemma (with $\widetilde{x} = \epsilon$).

Consider next the case where the antecedent is inferred through the rule for scope opening and $P = (\nu x)Q$. So we have $\alpha = \overline{a}(\nu x\widetilde{y})\widetilde{v}$, and in a shorter inference $Q\{\widetilde{w}/\widetilde{u}\} \xrightarrow{\alpha'} P'$ with $\alpha' = \overline{a}(\nu\widetilde{y})\widetilde{v}$, i.e., α extends α' by one additional binding in the object. There are several possibilities for R, but as in the previous case all of them can evolve (through one or two τ actions) to $(\nu\widetilde{z})(N \,|\, (\nu x')C')$ where $C' \in \mathcal{C}^{\widetilde{w}x'}_{\widetilde{u}x}(Q)$. Because bound names can be assumed to be distinct we can choose $x' = x$ and assume that x is not in $\widetilde{w}, \widetilde{u}$. So, $C' \in \mathcal{C}^{\widetilde{w}x}_{\widetilde{u}x}(Q)$. Therefore R' is related in $\mathcal{S}$ to $Q\{\widetilde{w}x/\widetilde{u}x\} = Q\{\widetilde{w}/\widetilde{u}\}$. Therefore, by induction and $Q\{\widetilde{w}/\widetilde{u}\} \xrightarrow{\alpha'} P'$ it holds that $(\nu\widetilde{z})(N \,|\, C')$ simulates Q as stated by the lemma. But then R also simulates P, through the scope opening rule.

The case where the antecedent is inferred through the rule for communication is notationally heavy so we will just go through the main ideas. If $P = P_1|P_2$ and $P \xrightarrow{\alpha} P'$ with $\alpha = \tau$ is inferred from P as a communication between P_1 and P_2, then we consider the output and input actions from P_1 and P_2 respectively; by induction they must be simulated by $(\nu\widetilde{z})(N \,|\, C_1)$ and $(\nu\widetilde{z})(N \,|\, C_2)$ where C_1 and C_2 belong to the proper $\mathcal{C}$-sets. We therefore have that $(\nu\widetilde{z})(N \,|\, C_1) \mid (\nu\widetilde{z})(N \,|\, C_2)$ can simulate the communication. The case is completed by showing that this agent is strongly equivalent with $(\nu\widetilde{z})(N \,|\, C_1 \,|\, C_2)$, which is in the $\mathcal{C}$-set for P itself. When the communication carries bound names restriction operators may appear in the derivatives. This explains the necessity of the "$(\nu\widetilde{x})$" in the statement of the lemma.

To show the strong equivalence needed for this case we establish that pairs

$$((\nu\tilde{z})(N \,|\, C_1) \;\mid\; (\nu\tilde{z})(N \,|\, C_2) \quad , \quad (\nu\tilde{z})(N \,|\, C_1 \,|\, C_2))$$

form a strong bisimulation. Clearly, the only difficulty in finding simulating transitions is for the left hand side to simulate a transition in the right hand side $(\nu\tilde{z})(N \,|\, C_1 \,|\, C_2)$ where C_1 and C_2 communicate along a trigger in $\tilde{z}$. Perhaps disappointingly, C_1 and C_2 may actually share such a trigger since the parallel composition may have been created by replication in an earlier part of the induction. Assume without loss of generality that the transition involves a trio in C_1 activating, along $z \in \tilde{z}$, a trio $z(\tilde{u}) \,.\, D$ in C_2. Then in C_1 there must also be a trio activated by z (since a trigger is only introduced in $\mathcal{B}$ along with a trio activated by it); moreover by the assumption about the uniqueness of the triggers and the fact that replication creates syntactically identical copies this trio must also be $z(\tilde{u}) \,.\, D$. So the transition then is of the kind

$$\begin{array}{l}(\nu\tilde{z})(N \;\mid\; C_1' \;\mid\; z(\tilde{u}) \,.\, D \;\mid\; C_2' \;\mid\; z(\tilde{u}) \,.\, D) \xrightarrow{\tau} \\ \qquad (\nu\tilde{z})(N \;\mid\; C_1'' \;\mid\; z(\tilde{u}) \,.\, D \;\mid\; C_2' \;\mid\; D\{\tilde{w}/\tilde{u}\})\end{array}$$

But then of course there is also another transition where C_1 activates its own trio rather than the one in C_2:

$$\begin{array}{l}(\nu\tilde{z})(N \;\mid\; C_1' \;\mid\; z(\tilde{u}) \,.\, D \;\mid\; C_2' \;\mid\; z(\tilde{u}) \,.\, D) \xrightarrow{\tau} \\ \qquad (\nu\tilde{z})(N \;\mid\; C_1'' \;\mid\; D\{\tilde{w}/\tilde{u}\} \;\mid\; C_2' \;\mid\; z(\tilde{u}) \,.\, D)\end{array}$$

The derivatives of these two transitions are strongly equivalent (since parallel composition is associative). The latter transition involves a communication within C_1 and can therefore be simulated by $(\nu\tilde{z})(N \,|\, C_1) \;\mid\; (\nu\tilde{z})(N \,|\, C_2)$, which therefore also simulates the former transition, up to strong equivalence.

Finally, consider the case where the antecedent is inferred through the rule for replication. Again we just provide a sketch. If $P = !\,Q$ and $P\{\tilde{w}/\tilde{u}\} \xrightarrow{\alpha} P'$, then through a shorter inference $(P|Q)\{\tilde{w}/\tilde{u}\} \xrightarrow{\alpha} P'$. Then we know by induction that this is simulated by $R = (\nu\tilde{z})(N|C)$ where $C \in \mathcal{C}_u^{\tilde{w}}(P|Q)$. Consider this C; it is (possibly after one or two τ-actions) a parallel composition $C_P|C_Q$ where the factors belong to the $\mathcal{C}$-sets of P and Q respectively. It then follows (by definition of $\mathcal{D}$ for replication: just increase m by one!) that this parallel composition is itself a member of $\mathcal{C}_u^{\tilde{w}}(P)$. Therefore the simulation of R for $P|Q$ also is an adequate simulation for P. $\square$

Lemma 7 *Assume* $P\{\widetilde{w}/\widetilde{u}\}\mathcal{S}R$. *If* $R \xrightarrow{\alpha} R'$, *then either (i)* $\alpha = \tau$ *and* $P\{\widetilde{w}/\widetilde{u}\}\mathcal{S} \sim R'$, *or (ii)* $P\{\widetilde{w}/\widetilde{u}\} \xrightarrow{\alpha} P'$ *and for some* $\widetilde{x}$, $P' \sim (\nu\widetilde{x})P''$ *and* $R' \sim (\nu\widetilde{x})R''$ *and* $P''\mathcal{S}R''$.

PROOF: In R we say that the *essential* prefixes are the ones with subject a, and also the τ-prefix, in the definitions of $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$. Thus these correspond directly to the prefixes in P. We distinguish the cases when $R \xrightarrow{\alpha} R'$ is inferred from the action from at least one essential prefix, and when it is inferred without any such actions.

Assume $R \xrightarrow{\alpha} R'$ is inferred without any actions from essential prefixes. Then it must be an internal communication within $R = (\nu\widetilde{z})(N \mid C)$ for $C \in \mathcal{C}^{\widetilde{w}}_{\widetilde{u}}(P)$, so $\alpha = \tau$. We can then show that also $P\{\widetilde{w}/\widetilde{u}\}\mathcal{S} \sim R'$. The proof is through a tedious inspection of the definitions, by showing that if $C \in \mathcal{C}^{\widetilde{w}}_{\widetilde{u}}(P)$ and $C \xrightarrow{\alpha} C'$ does not involve an essential prefix, then in this case also C' is strongly equivalent to a member of $\mathcal{C}^{\widetilde{w}}_{\widetilde{u}}(P)$. In other words, $\mathcal{C}^{\widetilde{w}}_{\widetilde{u}}(P)$ is closed under transitions not involving essential prefixes. This satisfies part (i) of the consequent of the lemma.

Then assume $R \xrightarrow{\alpha} R'$ involves an essential prefix. In this case we can show part (ii) of the consequent of the lemma. The proof is mainly a converse of the proof of Lemma 6, noting that when R has an unguarded essential prefix then the corresponding prefix in P also is unguarded. We omit the details which are long but routine given the proof of Lemma 6. □

We can now easily complete the proof of Theorem 3. From the three lemmata above it follows that $\mathcal{S}$ is a weak open bisimulation up to restriction and strong equivalence. Bisimulations up to restriction and equivalence have been treated extensively in previous work on the π-calculus [MPW92] and in a similar manner we conclude that agents related by $\mathcal{S}$ are weak open equivalent. In particular, $P \approx \mathcal{T}(P)$.

5 Conclusion

We have seen that concerts inherit all the expressive power of the full π-calculus, up to weak equivalence. The concerts are limited in that prefixes are nested to the depth of at most three, that parallel composition does not occur under prefix, that replication is only applied to trios, and that there is only one occurrence of restriction under replication. This type of

"minimal parallel components" is to our knowledge the only of its kind for the π-calculus or indeed for any process algebra with an asynchronous parallel operator (disallowing multiway rendezvous).

The consequences of this result are significant in the theory of the π-calculus. We still have no good measure on the expressiveness of the full calculus. Obtaining such a measure may involve demonstrating that other basic formalisms can encode the π-calculus and here it may be useful to know that it suffices to encode a limited form of the calculus, for example where parallel composition occurs only at a top level. Also, when designing implementation strategies for the calculus it may help to know that it suffices to implement limited forms, for example that one central name provider is enough.

The result can also be seen as a negative result in the quest for decidable subcalculi. It is known that in the full π-calculus it is possible to constructively encode arbitrary Turing machines. Therefore problems such as equivalence and model checking and termination of π-calculus terms are undecidable. The expressive power and hence undecidability can be seen as emanating from the replication operator, which is the only operator through which the size of a term can increase in a transition. Without replication the sets of transition sequences from terms are finite (up to alpha-conversion) and hence the problems mentioned above are decidable. But there are decidable subcalculi which admit terms with infinite sets of transition sequences; a prime example is the so called "finite-control" fragment investigated by Dam [Dam95] formed by using recursion rather than replication and requiring that parallel composition does not occur in recursive definitions. It may have been hoped that a similar decidable subcalculus can be found with replication rather than recursion, but this appears now not to be the case: even if replication is limited to trios the calculus is fully expressive and hence undecidable (since our translation into concerts is constructive). Since recursion and replication are interdefinable ($!P$ can be defined recursively as $!P \stackrel{\text{def}}{=} !P \,|\, P$, note that this is *not* finite-control) the result also puts a limit on the complexity of recursive definitions required for full expressiveness.

An obvious question is whether a similar result holds for *duos*, two nested prefixes, of kind $\alpha\,.\,\beta\,.\,\mathbf{0}$, in place of trios. The answer is negative: when replication is limited to duos, even some finite-control agents are not expressible! To see this consider the agent $P \stackrel{\text{def}}{=} a\,.\,b\,.\,P$ where for simplicity the prefixes

a, b have empty objects. (Using replication instead of recursion we would write $P = (\nu z)(\overline{z} \,|\, !z\,.\,a\,.\,b\,.\,\overline{z})$). Let D be an agent where replication only operates on duos and assume $D \approx P$. We shall derive a contradiction. For $D \approx P$ to hold there must be an infinite transition sequence

$$D \Longrightarrow D_0 \stackrel{a}{\longrightarrow} D_1 \stackrel{b}{\Longrightarrow} D_2 \stackrel{a}{\longrightarrow} D_3 \stackrel{b}{\Longrightarrow} \cdots$$

where $D_n \approx D_{n+2}$ and $D \approx D_0$. Each of the actions a in this sequence must emanate from some prefix in D. But D is syntactically finite so some prefix in D must give rise to infinitely many a. That prefix therefore must lie under a replication and hence in a duo in D. Consider some $D_{2n} \stackrel{a}{\longrightarrow} D_{2n+1}$ where such a prefix gives rise to the a action. There are two cases depending on where in the duo the prefix resides. Either the prefix is the first in the duo in D, as in $!a\,.\,\gamma\,.\,\mathbf{0}$. Then the same duo is present in D_{2n} (the operands of replication do not change as a result of transitions) and since it is replicated and can yield one a transition it can yield several, i.e. $D_{2n} \stackrel{a}{\longrightarrow} D' \stackrel{a}{\longrightarrow} \cdots$, contradicting $D_{2n} \approx D$. Or the prefix is the last in the duo in D, then D_{2n} must have formed from D by executing the first prefix in that duo leaving a parallel factor $a\,.\,\mathbf{0}$. In $D_{2n} \stackrel{a}{\longrightarrow} D_{2n+1}$ we then have that D_{2n+1} is formed by replacing the factor $a\,.\,\mathbf{0}$ by $\mathbf{0}$. Since $D_{2n+1} \stackrel{b}{\Longrightarrow}$ it then follows $D_{2n} \stackrel{b}{\Longrightarrow}$ because the addition of a parallel factor cannot decrease the possible transitions. But $D_{2n} \stackrel{b}{\Longrightarrow}$ contradicts $D_{2n} \approx D$. The conclusion is that no such D can exist.

Limiting replication to duos we can still define some non-finite-control processes (e.g., $!a\,.\,b\,.\,\mathbf{0}$ is not finite-control) so the decidability of equivalence in this situation is open. If we restrict attention to concerts of possibly replicated duos (forbidding agents like $a\,.\,!b\,.\,c\,.\,\mathbf{0}$) then there are even finite agents which are not expressible. For example, $a\,.\,b\,.\,a\,.\,\mathbf{0}$ is not expressible as a concert of duos; an argument similar to the above shows that b here can occur neither at the head nor at the tail of a duo. Combining such arguments we get the picture in Figure 1 exhibiting relative expressive power of some classes of agents.

If replication is limited to solos (one prefix) then agents are finite-control and equivalence is therefore decidable (the agent $!\alpha\,.\,\mathbf{0}$ is equivalent to the finite-control recursive definition $P \stackrel{\text{def}}{=} \alpha\,.\,P$).

In this paper we have used the polyadic π-calculus. A corresponding result appears impossible in the monadic π-calculus, where the objects in

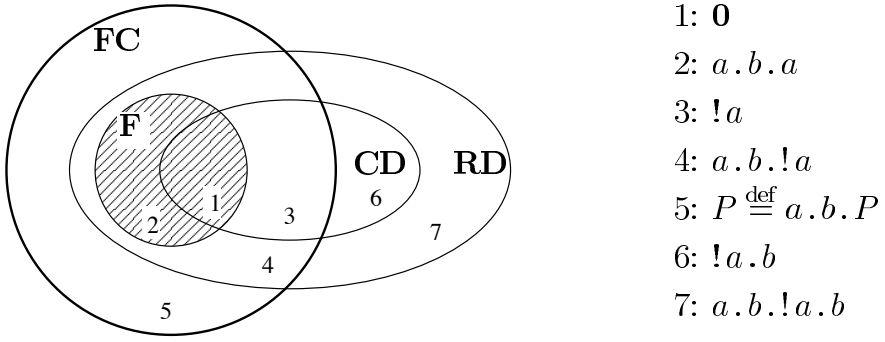

Figure 1: Relative expressiveness of the classes **FC** (finite-control), **F** (finite), **RD** (replication restricted to duos) and **CD** (concerts of possibly replicated duos). All regions are nonempty, the numbers show examples of inhabitants.

input and output actions must have length one, i.e., exactly one name is transferred in each communication. For consider the agent

$$a(x_1)\,.\,\cdots\,.\,a(x_n)\,.\,\overline{b}x_1\,.\,\cdots\,.\,\overline{b}x_n$$

If we want to translate this with a similar strategy where each subagent is enacted by starting a designated agent, then the designated agent for the subterm $\overline{b}x_1\,.\,\cdots\,.\,\overline{b}x_n$ must receive n names, and in the monadic calculus this requires a string of n input prefixes. However, by generalizing the notion of concert so that arbitrary strings of prefixes (rather than just trios) are allowed we can obtain a similar construction. This of course leaves open the question of expressiveness and decidability of concerts of trios in the monadic calculus.

Also, the construction is impossible for CCS, which can be thought of as the zero-ary π-calculus in that no objects are allowed in actions. The reason has here to do with the restriction operator. If restriction occurs under a replication operator then that entails a mechanism for creating new names. In the π-calculus concerts this mechanism is localized to a single name provider which sends those names, as objects in communications, to the places where they are needed. In CCS this is not possible so a similar strategy for encoding restriction will not work. Even relaxing the requirement that there can only be one inventive trio will not help; for example when encoding $!\,(\nu x)(P\,|\,Q)$ the encodings of P and Q must somehow learn of their new

common private name x. For this reason it appears that parallel compositions under replication is necessary in CCS, though again the question is open.

Finally, the result is not applicable in the presence of summation. Using unguarded summation, $P+Q$, means that weak equivalence is not a congruence and it is difficult to see how the translation could be extended. However with guarded summation, of the form

$$\alpha_1 \,.\, P_1 + \cdots + \alpha_n \,.\, P_n$$

I conjecture an extension which requires the use of *branch trios* of the kind $\alpha \,.\, (\beta + \gamma \,.\, \delta)$ (these contain four prefixes and only earn the attribute "trio" in that prefix is still nested to depth three). The translation of the guarded summation above is a parallel composition of one branch trio

$$z_i(\widetilde{u}) \,.\, (\overline{z_{i+1}}\widetilde{u} + \alpha_i \,.\, \overline{z_{P_i}}\widetilde{u})$$

for each summand $\alpha_i \,.\, P_i$ where addition in the index of z is modulo n. The parallel composition of the branch trios works like a round-robin protocol in a ring through the triggers z_i; the "token" $\widetilde{u}$ is passed around the ring until one of the trios decides on its leading action α_i. For finitely branching agents unguarded summation can be encoded with guarded summation so this device is quite powerful. Of course, using only proper concerts it is impossible to encode any kind of summation since $\alpha+\beta$ is not weakly equivalent to any summation-free agent. Similarly, instances of the *matching* operator $[x = y]P$ (read "if $x = y$ then P") from many version of the π-calculus cannot be encoded because $[x = y]\alpha$ is not equivalent to any matching-free agent.

References

[Bar84] H. Barendregt. *The Lambda Calculus*. North Holland, 1984.

[BW90] J. Baeten and W. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.

[Dam95] M. Dam. On the decidability of process equivalences for the π-calculus. In V.S. Alagar and M. Nivat, editors, *4th International Conference on Algebraic Methodology and Software Technology,*

AMAST'95, volume 936 of *Lecture Notes in Computer Science*, pages 169–183. Springer-Verlag, 1995.

[dS85] R. de Simone. Higher-level synchronising devices in MEIJE-SCCS. *Theoretical Computer Science*, 37(3):245–267, 1985.

[Gon85] G. Gonthier. Algebraic calculi of processes and net expressions. *Theoretical Computer Science*, 40:329–337, 1985.

[HT92] K. Honda and M. Tokoro. On asynchronous communication semantics. In M. Tokoro, O. Nierstrasz, and P. Wegner, editors, *Object-Based Concurrent Computing*, volume 612 of *Lecture Notes in Computer Science*, pages 21–51. Springer-Verlag, 1992.

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

[Mil91] R. Milner. The polyadic π-calculus: a tutorial. Technical Report ECS-LFCS-91-180, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, UK, October 1991. Also in *Logic and Algebra of Specification*, ed. F. L. Bauer, W. Brauer and H. Schwichtenberg, Springer-Verlag, 1993.

[Mil92] R. Milner. Functions as processes. *Journal of Mathematical Structures in Computer Science*, 2(2):119–141, 1992.

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

[OP92] F. Orava and J. Parrow. An algebraic verification of a mobile network. *Journal of Formal Aspects of Computing*, 4:497–543, 1992.

[Ora94] F. Orava. *On the Formal Analysis of Telecommunication Protocols*. PhD thesis, Department of Computer Systems, Uppsala University, Sweden, May 1994.

[Par90] J. Parrow. The expressive power of parallelism. *Future Generation Computer Systems*, 6:271–285, 1990.

[San93] D. Sangiorgi. From π-calculus to Higher-Order π-calculus — and back. In *Proceedings of TAPSOFT'93*, volume 668 of *Lecture Notes in Computer Science*, pages 151–166. Springer-Verlag, 1993.

[San94] D. Sangiorgi. On the bisimulation proof method. Revised version of Technical Report ECS–LFCS–94–299, University of Edinburgh, 1994. An extended abstract can be found in Proc. of MFCS'95, LNCS 969, 1994. Available electronically as `ftp://ftp.dcs.ed.ac.uk/pub/sad/bis-proof.ps.Z`.

[San96] D. Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Informatica*, 33:69–97, 1996. Earlier version published as Report ECS-LFCS-93-270, University of Edinburgh. An extended abstract appeared in LNCS 715 (Proc. CONCUR'93).

[Vaa93] F. Vaandrager. Expressiveness results for process algebras. In J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, *Proceedings REX Workshop on Semantics: Foundations and Applications,* Beekbergen, The Netherlands, June 1992, volume 666 of *Lecture Notes in Computer Science*, pages 609–638. Springer-Verlag, 1993.

[Wal95] D. Walker. Objects in the π-calculus. *Journal of Information and Computation*, 116(2):253–271, 1995.