Eike Best (Ed.)

CONCUR'93

4th International Conference
on Concurrency Theory
Hildesheim, Germany, August 23-26, 1993
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Series Editors

Gerhard Goos
Universität Karlsruhe
Postfach 69 80
Vincenz-Priessnitz-Straße 1
D-76131 Karlsruhe, Germany

Juris Hartmanis
Cornell University
Department of Computer Science
4130 Upson Hall
Ithaca, NY 14853, USA

Volume Editor

Eike Best
Institut für Informatik, Universität Hildesheim
Marienburger Platz 22, D-31141 Hildesheim, Germany

CR Subject Classification (1991): F.3, F.1, D.3

ISBN 3-540-57208-2 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-57208-2 Springer-Verlag New York Berlin Heidelberg

Typesetting: Camera-ready by authors
Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
45/3140-543210 - Printed on acid-free paper

Preface

CONCUR'93 is the fourth in an annual series of conferences devoted to the study of concurrency. It is a sequel of CONCUR'90 and CONCUR'91, both held in Amsterdam (The Netherlands), and CONCUR'92, held in New York (USA). The basic aim of the CONCUR conferences is to communicate advances in concurrency theories and applications.

This volume contains 31 papers that have been selected from 113 submissions; four invited papers and two abstracts of invited talks are also included. The members of the program committee and their subreferees have carefully read the submitted papers and selected the collection that is presented in this volume during a meticulous evaluation process. I would like to express my gratitude to them for their painstaking and valuable work.

I would also like to thank the organising committee for their efforts in arranging the conference, and Hildesheim University for hosting CONCUR'93.

Support for CONCUR'93 has generously been provided by the Deutsche Forschungsgemeinschaft and the Ministerium für Wissenschaft und Kultur des Landes Niedersachsen. The conference has also been supported by the Commission of the European Communities (Esprit Basic Research), SUN Microsystems, IBM, Technologietransferstelle der Universität Hildesheim, Stadtsparkasse Hildesheim, Pfeffer Bürotechnik and Frings & Kuschnerus. The National Science Foundation has supported the conference by providing a number of travel grants for US academic researchers and graduate students.

Hildesheim, July 1993 — Eike Best

Organising Committee

Ursula Goltz (Chair)
Vera Doering
Sabine Karmrodt
Bernd Grahlmann
Michaela Huhn
Ernst-Rüdiger Olderog (Tutorials Chair)

Invited Speakers

Jan A. Bergstra
Bard Bloom
Gérard Boudol
Cliff B. Jones
Christian Lengauer
Pierre Wolper

Program Committee

Jos C.M. Baeten
Manfred Broy
Raymond Devillers
Matthew Hennessy
Mogens Nielsen
Colin Stirling
David B. Benson
Ilaria Castellani
Javier Esparza
Bengt Jonsson
Catuscia Palamidessi
P.S. Thiagarajan
Walter Vogler
Eike Best (Chair)
W. Rance Cleaveland
Roberto Gorrieri
Maciej Koutny
Doron Peled
Frits W. Vaandrager

CONCUR'93 Subreferees

Wil van der Aalst
Parosh Abdulla
Luca Aceto
Rajeev Alur
Davide Ancona
Henrik R. Andersen
Andrea Asperti
Eric Badouel
Dieter Barnard
Javier Blanco
Bard Bloom
Frank de Boer
Roland Bol
Doeko Bosscher
Gérard Boudol
Martin Bonsangue
Frédéric Boussinot
Julian Bradfield
Franck van Breugel
Hans H. Brüggemann
Glenn Bruns
Nadia Busi
Ufuk Celikkan
Antonio Cerone
Allan Cheng
Ivan Christoff
Linda Christoff
Paolo Ciancarini
Ricardo Civalero
Jos Coenen
Gerardo Costa
Ruggero Costantini
Mads Dam
Philippe Darondeau
Jörg Desel
Volker Diekert
Eugeniusz Eberbach
Juan V. Echagüe
Herbert Ehler
Uffe Engberg
Urban Engberg
C. Facchi
Gian Luigi Ferrari
Thomas Filkorn
Willem Jan Fokkink
Lars-åke Fredlund
David de Frutos
Max Fuchs
Netty van Gasteren
Rob Gerth
Rob van Glabbeek
Stefania Gnesi
Robert Gold
Dominik Gomm
Susanne Graf
Thomas Gritzner
Jan Friso Groote
Orna Grumberg
Alessio Guglielmi
Jeremy Gunawardena
Carl Gunter
Elsa Gunter
Zineb Habbas
Hans Hansson
David A. Harrison
Kees van Hee
Andreas Heise
Yoram Hirshfeld
Gerard Holzmann
Jozef Hooman
Richard P. Hopkins
Chris Ho-Stuart
Huimin Lin
Kees Huizing
H. Hußmann
Hans Hüttel
Ryszard Janicki
Matthias Jantzen
Alan Jeffrey
Claus T. Jensen
Roope Kaivola
Shmuel Katz
Astrid Kiehn
Ekkart Kindler
Nils Klarlund
Steven Klusener
Henri Korver
Ruurd Kuiper
Robert P. Kurshan
Cosimo Laneve
F. Laroussinie
Kim G. Larsen
Peter Ernst Lauer
Hans-G. Linde-Göers
Kamal Lodaya
Angelika Mader
Thierry Massart
Sjouke Mauw
Michael Merritt
Giorgio De Michelis
Bernhard Möller
Eugenio Moggi
Faron Moller
Ugo Montanari
Peter D. Mosses
Ben Moszkowski
Madhavan Mukund
Hans Mulder
David Murphy
V. Natarajan
Dieter Nazareth
Rocco De Nicola
Xavier Nicollin
Flemming Nielson
Fredrik Orava
Yolanda Ortega
Barbara Paech
Giuseppe Pappalardo
Joachim Parrow
Elisabeth Pelz
Ricardo Peña
Adriano Peron
Sophie Pinchinat
S. Purushothaman
Alexander Rabinovich
R. Ramanujam
Paul Rambags
Gianna Reggio
Wolfgang Reisig
John Reppy
Jon Riecke
James Riely
Jan Rutten
Davide Sangiorgi
Vladimiro Sassone
Bernhard Schätz
Oliver Schoett
Glauco Siliprandi
Carla Simone
Robert de Simone
Steve Sims
Antonia Sinachopoulos
Scott Smolka
Sergei Soloviev
Katarina Spies
Eugene Stark
Iain A. Stewart
Ketil Stoelen
Frank Stomp
Gadi Taubenfeld
Chris Tofts
David Turner
Chris Verhoef
Björn Victor
Fer-Jan de Vries
Rolf Walter
Jos van Wamel
Rainer Weber
Glynn Winskel
Xinxin Liu
Alex Yakovlev
Daniel Yankelevich
Wang Yi
Hussein S.M. Zedan
Wiesław Zielonka
Zhenhua Duan
Elena Zucca
Jeffrey Zucker

Table of Contents

The Lambda-Calculus with Multiplicities

(abstract)

Gérard Boudol

INRIA-Sophia Antipolis

BP 93, 06902 SOPHIA-ANTIPOLIS CEDEX (FRANCE)

email: gbo@sophia.inria.fr

The λ-*calculus with multiplicities* is a refinement of the usual λ-calculus, inspired by the encoding of the lazy λ-calculus into the π-calculus given by Milner in [Milner 1992]. The basic observation is this: in a reduction step $(\lambda x.M)N \rightarrow M[N/x]$, the argument N is copied as many times as we need, that is, as much as there are free occurrences of x in M. One could say that N is infinitely available. On the other hand, the π-calculus provides means, namely parallel composition and replication (or "bang"), for controlling the number of copies of an agent. One can show that this allows for distinguishing terms that are not distinguished in the λ-calculus.

In the refinement of the λ-calculus we propose, the argument of a function is a *bag of resources*, that is more precisely a multiset of terms. Then each term in the bag comes with an explicit, possibly infinite *multiplicity*, indicating how many copies of it are available. One recovers the usual λ-calculus when the bags consist of just one term with an infinite multiplicity. We shall write a bag as a parallel composition $P = (M_1^{m_1} \mid \cdots \mid M_k^{m_k})$ of terms with multiplicities (where m_i is a non-negative integer, or ∞). The parallel composition is intended to be commutative and associative, with a neutral element $\mathbf{1}$, denoting the empty multiset. Then, besides the variables x, y, $z\ldots$ and the abstractions $\lambda x.M$, the syntax of our calculus includes applications of the form (MP), where M is any term, and P a bag of terms. The management of the resources is done by means of explicit substitutions. That is, we use $M[P/x]$ not just as a notation for the meta-operation of substitution, but as

---

This work has been partly supported by the BRA CONFER, and by the PRC-CNRS "Modèles Logiques de la Programmation".

an explicit syntactic contruct, as in [Abadi et al. 1990], which binds the variable x in M. Summarizing, the syntax of the λ-calculus with multiplicities is as follows:

$$M ::= x \mid \lambda x.M \mid (MP) \mid (M[P/x])$$
$$P ::= \mathbf{1} \mid M \mid (P \mid P) \mid M^\infty$$

The terms with finite multiplicity may be defined by:

$$M^0 = \mathbf{1}$$
$$M^{m+1} = (M \mid M^m)$$

The usual λ-calculus is obtained by restricting the syntax of bags of resources to $P ::= M^\infty$.

The evaluation rules are given in the appendix. The evaluation process is basically the lazy β-reduction, but we also have to incorporate the substitution process as a part of the evaluation. To evaluate $M[P/x]$, one fetches a resource from the bag P, if any, when one actually needs it, that is when the variable x is in the head position in M. Typically, we have:

$$(xP_1 \dots P_n)[(M \mid P)/x] \to (MP_1 \dots P_n)[P/x]$$

The *operational semantics* of our calculus is defined as the usual Morris' preorder, relying on the notion of *value*. A value is any functional closure, that is any term given by the grammar:

$$V ::= \lambda x.M \mid (V[P/x])$$

We write $M\Downarrow$ whenever M has a value, that is $\exists V.\ M \to^* V$. Then the operational preorder is given by:

$$M \sqsubseteq N \ \Leftrightarrow_{\text{def}}\ \forall C.\ \ C[M]\Downarrow\ \Rightarrow\ C[N]\Downarrow$$

where C denotes any context (a term with a hole) and $C[M]$ is the term obtained by placing M in the hole of C.

It is worth emphasizing two points here. First, if the bag P in $M[P/x]$ is empty, nothing can be fetched out of it. Then a term like $x[\mathbf{1}/x]$ is *deadlocked*. It is a closed normal form w.r.t. the evaluation process, but not a value. Second, since parallel composition is commutative and associative, any resource from the bag can be selected in the fetch operation. Then we can define a *non-deterministic choice* $(M \oplus N)$ as follows, provided x is not free in M or N:

$$(M \oplus N) =_{\text{def}} x[(M \mid N)/x]$$

One can see that $(M \oplus N) \to M[N/x]$ and $(M \oplus N) \to N[M/x]$. Moreover, the two terms $M[N/x]$ and $N[M/x]$ may be regarded as identical to M and N respectively, up to some garbage collection. These two features of deadlock and non-determinism do not arise in the usual λ-calculus. The non-deterministic choice allows one to deal with parallel functions, as in [Boudol 1990] (see also [Boudol 1991]).

One can show that the introduction of explicit finite multiplicities strictly increases the discriminating power of the λ-calculus. For instance, the two λ-terms $A = (xx^\infty)$ and $B = x(\lambda y.(xy^\infty))^\infty$ cannot be operationally distinguished in the λ-calculus. In fact, they are equated in any model satisfying a weak form of extensionality, namely the conditional η-rule $M \neq \Omega \;\;\Rightarrow\;\; M = \lambda x(Mx^\infty)$, for x not free in M. In our calculus, they are distinguished by the context $C = [.][\lambda z.z/x]$, since we have:

$$C[A] \to ((\lambda z.z)x^\infty)[\mathbf{1}/x] \to z[x^\infty/z][\mathbf{1}/x] \to x[x^\infty/z][\mathbf{1}/x]$$

which is deadlocked, while

$$\begin{aligned} C[B] \to ((\lambda z.z)(\lambda y.(xy^\infty))^\infty)[\mathbf{1}/x] &\to z[(\lambda y.(xy^\infty))^\infty/z][\mathbf{1}/x] \\ &\to \lambda y.(xy^\infty)[(\lambda y.(xy^\infty))^\infty/z][\mathbf{1}/x] \end{aligned}$$

that is a value. The same context can be used to separate the two terms of Ong's example, see [Milner 1992]. Moreover, λ-terms which are separated by means of parallel functions can be operationally distinguished using the non-deterministic choice, see [Boudol 1990].

Not surprisingly, the λ-calculus with multiplicities bears some relationship with Girard's Linear Logic [Girard 1987]. More precisely, we show that a fragment of linear logic provides a suitable *functionality theory*, in the sense of Curry and Coppo (see [Coppo et al. 1981]), for our calculus. We do *not* use our calculus to develop a proof theory for linear logic, i.e. a Curry-Howard correspondence, as Abramsky does with his linear calculus in [Abramsky 1993]. We merely give a semantics for our calculus using "linear formulae", and then we slightly modify Girard's original concrete syntax. Our fragment of linear logic is given by:

$$\begin{aligned} \phi &::= \omega \mid (\phi \wedge \phi) \mid (\pi \multimap \phi) \\ \pi &::= \mathbb{1} \mid \phi \mid (\pi \times \pi) \mid\ !\phi \end{aligned}$$

where ω stands for "additive truth" ($\top$), $\wedge$ is the additive conjunction ($\&$), $\multimap$ is the (linear) implication, $\mathbb{1}$ the "multiplicative truth", $\times$ is the multiplicative conjunction ($\otimes$), and ! is the exponential "of course". Formulae of the first kind (ϕ) provide functional characters for the terms of the calculus, while the formulae of the second kind (π) are used to "type" multisets of resources.

The typing rules are given in the appendix. One may observe that if we forget the terms, we get exactly the rules of the corresponding fragment of linear logic. Then our main result is a refinement of a theorem by Coppo, Dezani and Venneri [Coppo et al. 1981], relating the convergence of the evaluation process on a given term with the fact that this term has a non-trivial functional character. As a matter of fact, this result asserts that the functionality theory provides an *adequate semantics* for our λ-calculus with multiplicities. This is stated formally in the appendix.

REFERENCES

[Abadi et al. 1990] M. ABADI, L. CARDELLI, P.-L. CURIEN, J.-J. LÉVY, *Explicit substitutions*, POPL 90 (1990) 31-46.

[Abramsky 1993] S. ABRAMSKY, *Computational interpretations of linear logic*, Theoretical Comput. Sci. 111 (1993) 3-57.

[Boudol 1990] G. BOUDOL, *A λ-calculus for parallel functions*, INRIA Res. Report 1231 (1990).

[Boudol 1991] G. BOUDOL, *Lambda-calculi for (strict) parallel functions*, INRIA Res. Report 1387 (1991) to appear in Information and Computation.

[Coppo et al. 1981] M. COPPO, M. DEZANI-CIANCAGLINI, B. VENNERI, *Functional characters of solvable terms*, Zeit. Math. Logik Grund. 27 (1981) 45-58.

[Girard 1987] J.-Y. GIRARD, *Linear Logic*, Theoretical Comput. Sci. 50 (1987) 1-102.

[Milner 1992] R. MILNER, *Functions as processes*, Math. Struct. in Comp. Science 2 (1992) 119-141.

Appendix: Evaluation and the Functionality System.

We denote by $M =_\alpha N$ the α-conversion, and by $P \equiv Q$ the structural equivalence of bags of terms, defined as the least congruence (on "bags") satisfying:

$$\begin{aligned}(\mathbf{1} \mid P) &\equiv P\\ (P \mid Q) &\equiv (Q \mid P)\\ (P \mid (Q \mid R)) &\equiv ((P \mid Q) \mid R)\\ M^\infty &\equiv (M \mid M^\infty)\end{aligned}$$

This is extended to terms as the least congruence such that:

$$P \equiv Q \ \Rightarrow\ MP \equiv MQ \quad \& \quad M[P/x] \equiv M[Q/x]$$

The evaluation relation $M \to M'$ uses an auxiliary relation $M[N/x] \mapsto M'$, the "fetch" relation. These are inductively given as follows:

$$\frac{M \to M'}{N \to M'}\ M =_\alpha N \qquad\qquad \frac{M \to M'}{N \to M'}\ M \equiv N$$

$$\frac{M \to M'}{MP \to M'P} \qquad\qquad \frac{M \to M'}{M[P/x] \to M'[P/x]}$$

$$(\lambda x.M)P \to M[P/x] \qquad \frac{(VP) \to M}{(V[Q/x])P \to M[Q/x]}\ x \text{ not free in } P$$

$$x[M/x] \mapsto M \qquad \frac{M[N/x] \mapsto M'}{(MP)[N/x] \mapsto M'P}$$

$$\frac{M[N/x] \mapsto M'}{(M[P/z])[N/x] \mapsto M'[P/z]}\ z \neq x,\ z \text{ not free in } N$$

$$\frac{M[N/x] \mapsto M'}{M[(N \mid P)/x] \to M'[P/x]}\ x \text{ not free in } N$$

The functionality sytem is an intuitionistic natural deduction system, presented in sequent form. The sequents are either $x_1 : \pi_1, \cdots x_k : \pi_k \vdash M : \phi$, for typing terms, or $x_1 : \pi_1, \cdots x_k : \pi_k \vdash P : \pi$, for typing bags of resources. As usual $!\Gamma$ denotes an hypothesis of the form $x_1 : !\phi_1, \cdots x_k : !\phi_k$. To factorize all the structural rules, which do not correspond to any term construct, we write $\Gamma \gg \Delta$ whenever Δ results from Γ by application of a sequence of exchange, weakening, dereliction, contraction or product. That is, $\gg$ is the least preorder satisfying:

$\Gamma, x : \sigma, y : \tau, \Delta \gg \Gamma, y : \tau, x : \sigma, \Delta$	*exchange*
$\Gamma \gg x : !\psi, \Gamma$	*weakening*
$\Gamma \gg x : \mathbb{1}, \Gamma$	*weakening*
$x : \psi, \Gamma \gg x : !\psi, \Gamma$	*dereliction*
$x : !\psi, x : !\psi, \Gamma \gg x : !\psi, \Gamma$	*contraction*
$x : \sigma, x : \tau, \Gamma \gg x : \sigma \times \tau, \Gamma$	*product*

The first group of typing rules concerns the constructions of the calculus:

$$x : \phi \vdash x : \phi \qquad \frac{\Gamma \vdash P : \pi\ ,\ x : \pi, \Delta \vdash M : \phi}{\Gamma, \Delta \vdash M[P/x] : \phi}\ (x \text{ not in } \Delta)$$

$$\frac{x : \pi, \Gamma \vdash M : \phi}{\Gamma \vdash \lambda x.M : \pi \multimap \phi}\ (x \text{ not in } \Gamma) \qquad \frac{\Gamma \vdash M : \pi \multimap \phi\ ,\ \Delta \vdash P : \pi}{\Gamma, \Delta \vdash (MP) : \phi}$$

$$\frac{\Gamma \vdash P : \sigma\ ,\ \Delta \vdash Q : \tau}{\Gamma, \Delta \vdash (P \mid Q) : \sigma \times \tau} \qquad \frac{!\Gamma \vdash M : \phi}{!\Gamma \vdash M^\infty : !\phi}$$

The remaining typing rules are independent from the structure of the terms:

$$\Gamma \vdash M : \omega \qquad\qquad \vdash P : \mathbb{1}$$

$$\frac{\Gamma \vdash T : \tau}{\Delta \vdash T : \tau}\ \Gamma \gg \Delta \qquad\qquad \frac{\Gamma \vdash M : \phi\ ,\ \Gamma \vdash M : \psi}{\Gamma \vdash M : \phi \wedge \psi}$$

$$\frac{\Gamma \vdash M : \phi \wedge \psi}{\Gamma \vdash M : \phi} \qquad\qquad \frac{\Gamma \vdash M : \phi \wedge \psi}{\Gamma \vdash M : \psi}$$

DEFINITION (INTERPRETATION). *$\mathcal{F}[\![M]\!]$ is the set of pairs (Γ, ϕ) such that $\Gamma \vdash M : \phi$, and*

$$N \sqsubseteq_{\mathcal{F}} M \ \Leftrightarrow_{\text{def}}\ \mathcal{F}[\![N]\!] \subseteq \mathcal{F}[\![M]\!]$$

THEOREM (ADEQUACY).

(i) *for M closed:* $M\Downarrow \ \Leftrightarrow\ \exists \pi, \phi.\ \vdash M : \pi \multimap \phi$

(ii) $\mathcal{F}[\![N]\!] \subseteq \mathcal{F}[\![M]\!] \ \Rightarrow\ N \sqsubseteq M$

A Multiset Semantics for the pi-Calculus with Replication

Joost Engelfriet*

Department of Computer Science, Leiden University
P.O.Box 9512, 2300 RA Leiden, The Netherlands

Abstract. A multiset (or Petri net) semantics is defined for the π-calculus with replication. The semantic mapping is a strong bisimulation, and structurally congruent processes have the same semantics.

1 Introduction

The π-calculus has recently been introduced as an extension of CCS to mobile concurrent processes (see [13, 10, 11, 12]). As for CCS [9], the (interleaving) semantics of the π-calculus is given by a transition system of which the states are process terms. In this paper we provide a Petri net semantics for the "small π-calculus", i.e., the subset of the π-calculus defined by Milner in [10]. The main features of this subset are that it has no choice operator and that recursion is replaced by the more elementary operation of replication, denoted by an exclamation mark: if P is a process, then $!P$ stands for a countably infinite number of concurrent copies of P. It is shown in [10] that this subset suffices to simulate important aspects of the λ-calculus.

Petri net semantics of process algebras has been studied in, e.g., [7, 14, 5, 17, 15]. In such a semantics, a Petri net is associated with each process; the idea is that this Petri net expresses the concurrency present in the process in a more direct way than the interleaving transition system. Here we wish to stress that a Petri net (and in particular a P/T net) is just a particular kind of transition system, viz. one of which the states are multisets (also called markings) and the transition relation $\rightarrow$ satisfies the following "chemical law" (where S_1, S_2, and S are states and $\cup$ is multiset union): if $S_1 \rightarrow S_2$, then $S_1 \cup S \rightarrow S_2 \cup S$. For this reason, we suggest the term "multiset transition system" as an alternative to "Petri net" (just as a "transition system" used to be called "automaton"). It has been the early insight of Petri that the (multi)set is exactly the datastructure that fits to the notion of concurrency, and that communication between elements of the (multi)set can be modeled by (multi)set replacement, as formalized by the chemical law. The suggestive "chemical" terminology is from [1, 3] where a multiset is viewed as a chemical soup of molecules (but we will use Petri nets rather than the recent CHemical Abstract Machine of [3], which has some unnecessary features and has been less well studied).

* The research of the author was supported by the Esprit Basic Working Group No.6067 CALIBAN.

The semantics of the "small π-calculus" is presented in [10] (and in [11]) in a novel way, inspired by the CHAM. First a so-called structural congruence is defined on the process terms that is meant to capture the fact that two processes are structurally, i.e., statically, the same. In other words, the processes have the same flow graph (see [9, 13]), which roughly means that they can be decomposed into the same concurrent subprocesses. Then, an interleaving transition system is given in which structurally congruent processes are given the same behaviour, by definition. This separation of "physical" structure and behaviour is intuitively clear, and simplifies the transition system to a large extent. In particular, the commutativity and associativity of parallel composition are handled on the structural level, and replication is even handled completely at the structural level (reducing it to parallel composition).

In this paper we wish to put forward the general idea that the multiset (or Petri net) semantics of a process algebra should also be used to express the structure of the processes: if two processes have the same structure, they should also have the same multiset semantics. Ideally we even would like two processes to have the same structure if and only if they have the same multiset semantics. Intuitively, the syntax of process terms that is needed to describe a multiset of concurrent subprocesses, should not be present in that multiset; the syntactic laws needed to describe multisets should in fact be sound, and preferably even complete.

We define one big multiset transition system (or Petri net), called Mπ, and we define a (compositional) semantic mapping that associates a state of Mπ with each process of the small π-calculus. Thus, the meaning of a process is a multiset (or marking of the net Mπ); intuitively, it is the multiset of all its concurrent subprocesses. The Petri net Mπ has one type of transition only, which corresponds to the basic action in the small π-calculus: a communication between two processes. In this way Mπ is similar to the "object-oriented" interleaving transition system of the small π-calculus. Our main results on this semantics are:

(A) the semantic mapping is a strong bisimulation between the interleaving transition system of the small π-calculus and the multiset transition system Mπ, and

(B) if two processes of the small π-calculus are structurally congruent, then they have the same semantics in Mπ.

Result (A) ensures that a process and its corresponding multiset in Mπ have the same (interleaving) behaviour. Result (B) means that two processes that have the same structure also have the same multiset semantics. The converse of (B) does not hold and thus the laws of structural congruence of the small π-calculus are sound, but not complete relative to the multiset semantics. We suggest that the structural congruence should be extended in such a way that (B) does hold in both directions; how to do this is open.

Our semantics satisfies, in a certain sense, the two requirements for a Petri net semantics to be a "good" semantics as formulated by Olderog in [15]. The first requirement is that the interleaving semantics should be "retrievable" from the Petri net semantics in the sense that they should be strongly bisimilar; this is exactly result (A). The second requirement is that the Petri net semantics should reflect the "intended concurrency". Although its formalization in [15] is not applicable here, we believe intuitively that it is fulfilled. The two CHAMs proposed in [3] for the small π-calculus both fail to satisfy the first requirement, due to their heating rules. In our opinion, they also fail to satisfy the second requirement, the first CHAM because it uses cooling rules to implement α-conversion, and the second CHAM because it has a non-distributed name server. The second CHAM is strongly related to our multiset semantics; in fact, our semantic mapping may be viewed as a one-stroke implementation of its heating rules.

The semantic mapping associates a multiset S in Mπ with each process term P. Intuitively, S is the decomposition of P into its concurrent subprocesses. Thus, the semantic mapping is similar to the decomposition mappings of [5, 15]; however, as opposed to [5, 15], it also decomposes all (guarded) subterms of P. Another difference is that it decomposes into multisets rather than sets; in fact, the replication operation forces us to consider non-safe Petri nets. The advantages of non-safe nets have been pointed out in [7]; runs (or "processes") of such nets have been studied in [8] (see also [6]). A final, essential, difference with [5, 15] is that it is impossible to reconstruct P from S; in fact, such a reconstruction would contradict the desired result (B).

2 The Small π-Calculus

We briefly recall the definition of the small π-calculus from [10].

Let N be an infinite set of names. The context-free syntax for process terms is as follows (where we use a comma rather than | to separate alternatives):

$$P \ ::= \ \overline{x}y.P \ , \ x(y).P \ , \ \mathbf{0} \ , \ P \mid P \ , \ !P \ , \ (\nu y)P$$

with $x, y \in$ N. The strings $\overline{x}y$ and $x(y)$ are called *guards*. The y in $x(y).P$ and in $(\nu y)P$ binds all free occurrences of y in P. We denote by fn(P) the set of names that occur free in process P; thus, fn(P) $\subseteq$ N.

Informally, process $\overline{x}y.P$ sends the name y along the link x and then continues as process P, and process $x(y).P$ receives any name z along the link x and then continues as process $P[z/y]$, where $P[z/y]$ denotes the result of substituting z for all free occurrences of y in P (renaming bound names where necessary, as usual). Parallel composition of processes P and Q is denoted $P \mid Q$ as usual in CCS, and $\mathbf{0}$ is the inactive process; $(\nu y)P$ is the restriction of y to P, denoted $P\backslash y$ in CCS. Finally, the process $!P$ is the replication of process P and abbreviates $P \mid P \mid P \mid \cdots$.

Structural congruence, denoted $\equiv$, is the smallest congruence over the set of all process terms such that

1. $P \equiv Q$ whenever P and Q are α-convertible,