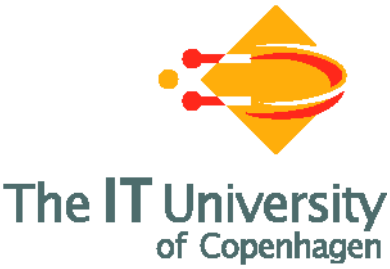


Preliminary Proceedings
14th International Workshop on Expressiveness in Concurrency

EXPRESS'07
Lisbon, Portugal
3 September 2007

Roberto Amadio (Université Paris 7)
Thomas Hildebrandt (IT University of Copenhagen)
editors

IT University Technical Report Series **TR-2007-100**

ISSN 1600–6100 **August 2007**

ISSN 1600–6100

ISBN 978-87-7949-157-1

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone: +45 72 18 50 00
Telefax: +45 72 18 50 01
Web www.itu.dk

Contents

Preface

The EXPRESS workshops aim at bringing together researchers interested in the relations between various formal systems, particularly in the field of Concurrency. More specifically, they focus on the comparison between programming concepts (such as concurrent, functional, imperative, logic and object-oriented programming) and between mathematical models of computation (such as process algebras, Petri nets, event structures, modal logics, rewrite systems etc.) on the basis of their relative expressive power.

The EXPRESS workshops were originally held as meetings of the HCM project EXPRESS, which was active with the same focus from January 1994 till December 1997. The first three workshops were held respectively in Amsterdam (1994, chaired by Frits Vaandrager), Tarquinia (1995, chaired by Rocco De Nicola), and Dagstuhl (1996, co-chaired by Ursula Goltz and Rocco De Nicola). EXPRESS'97, which took place in Santa Margherita Ligure and was co-chaired by Catuscia Palamidessi and Joachim Parrow, was organized as a conference with a call for papers and a significant attendance from outside the project. EXPRESS'98, co-chaired by Ilaria Castellani and Catuscia Palamidessi, was held as a satellite workshop of the CONCUR'98 conference in Nice and like on that occasion EXPRESS'99, co-chaired by Ilaria Castellani and Björn Victor, was hosted by the CONCUR'99 conference in Eindhoven. The EXPRESS'00 workshop, co-chaired by Luca Aceto and Björn Victor, was held as a satellite workshop of CONCUR 2000, Pennsylvania State University, USA. The EXPRESS'01 workshop, co-chaired by Luca Aceto and Prakash Panangaden, was held at BRICS, Aalborg University as a satellite of CONCUR'01. The EXPRESS'02 workshop, co-chaired by Uwe Nestmann and Prakash Panangaden, was held at Brno University as a satellite of CONCUR'02. The EXPRESS'03 workshop, co-chaired by Flavio Corradini and Uwe Nestmann, was co-located with CONCUR 2003, Marseille, France. The EXPRESS '04 workshop, co-chaired by Jos Baeten and Flavio Corradini, was co-located with CONCUR 2004, London, Great Britain. The EXPRESS '05 workshop, co-chaired by Jos Baeten and Iain Phillips, was co-located with CONCUR 2005, San Francisco, USA. The EXPRESS '06 workshop, co-chaired by Roberto Amadio and Iain Phillips, was co-located with CONCUR 2006, Bonn, Germany.

This year EXPRESS is co-located with CONCUR '07, held in Lisbon. In response to the call for papers, we received one short paper and 13 full papers. The program committee selected 4 of the full papers for presentation at the workshop. In addition, the workshop contains four invited presentations, by Cédric Fournet (invited jointly with the SECCO workshop), Jos Baeten, François Laroussinie, and Iain Phillips (invited jointly with the GTVC workshop), and a discussion introduced and moderated by Daniele Gorla and Uwe Nestmann on the definition of the criteria that make an encoding 'good'. Abstracts for the invited talks and discussion appear in these preliminary proceedings. We would like to thank the authors of the submitted papers, the invited speakers, the members of the program committee, and their subreferees for their contribution to both the meeting and this volume. Also we thank Daniele and Uwe for introducing the discussion, the CONCUR organising committee for hosting EXPRESS07, and the workshop organisers Francisco Martins and António Ravara for arranging the printing of these preliminary proceedings, which were compiled by Espen Højsgaard. The final proceedings will become available electronically at Elsevier's web site `http://www.elsevier.com/locate/entcs`.

The editors

Roberto Amadio (Universit Paris 7)
Thomas Hildebrandt (IT University of Copenhagen)

EXPRESS 2007 Program Committee

Roberto Amadio (co-chair)
Luis Caires
Sybille Froeschle
Philippa Gardner
Daniele Gorla
Thomas Hildebrandt (co-chair)
Bas Luttik
Uwe Nestmann
Julian Rathke
Roberto Segala
Pawel Sobocinski

EXPRESS 2007 Subreferees

Suzana Andova
Ingo Brueckner
Stefano Guerrini
Sergio Maffeis
Lutz Schröder
Daniele Varacca
Björn Victor

A Type Discipline for Authorization in Distributed Systems

Invited talk

Cédric Fournet[1,2]

Microsoft Research Ltd.
7 J J Thomson Avenue
Cambridge CB3 0FB
United Kingdom

Abstract

We consider the problem of statically verifying the conformance of the code of a system to an explicit authorization policy. In a distributed setting, some part of the system may be compromised, that is, some nodes of the system and their security credentials may be under the control of an attacker. To help predict and bound the impact of such partial compromise, we advocate logic-based policies that explicitly record dependencies between principals. We propose a conformance criterion, "safety despite compromised principals", such that an invalid authorization decision at an uncompromised node can arise only when nodes on which the decision logically depends are compromised. We formalize this criterion in the setting of a process calculus, and present a verification technique based on a type system. Hence, we can verify policy conformance of code that uses a wide range of the security mechanisms found in distributed systems, ranging from secure channels down to cryptographic primitives, including secure hashes, encryption, and public-key signatures.

[1] Joint work with Andrew Gordon and Sergio Maffeis.

[2] Email: fournet@microsoft.com

This is a preliminary version. The final version will be electronically published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.com/locate/entcs

Expressing Priorities and External Probabilities in Process Algebra via Mixed Open/Closed Systems

Mario Bravetti[1,2]

Dipartimento di Scienze dell'Informazione
Università di Bologna
Bologna, Italy

Abstract

Defining operational semantics for a process algebra is often based either on labeled transition systems that account for interaction with a context or on the so-called reduction semantics: we assume to have a representation of the whole system and we compute unlabeled reduction transitions (leading to a distribution over states in the probabilistic case). In this paper we consider mixed models with states where the system is still open (towards interaction with a context) and states where the system is already closed. The idea is that (open) parts of a system "P" can be closed via an operator "$P \uparrow G$" that turns already synchronized actions whose "handle" is specified inside "G" into prioritized reduction transitions (and, therefore, states performing them into closed states). We show that we can use the operator "$P \uparrow G$" to express multi-level priorities and external probabilistic choices (by assigning *weights* to handles inside G), and that, by considering reduction transitions as the only unobservable τ transitions, the proposed technique is compatible, for process algebra with general recursion, with both standard (probabilistic) observational congruence and a notion of equivalence which aggregates reduction transitions in a (much more aggregating) trace based manner. We also observe that the trace-based aggregated transition system can be obtained directly in operational semantics and we present the "aggregating" semantics. Finally, we discuss how the open/closed approach can be used to also express discrete and continuous (exponential probabilistic) time and we show that, in such timed contexts, the trace-based equivalence can aggregate more with respect to traditional lumping based equivalences over Markov Chains.

Keywords: Process algebra, Priorities, Probabilities, Congruence property.

1 Introduction

In the literature, two main approaches are commonly used to define the semantics of a process algebra in an operational way. The first one, originally used to define the sematics of CCS [6], is based on *labeled* transition systems: the labels are used to represent both internal behaviours and possible behaviors obtained by interacting with a context. In the following we will refer to such labeled transition systems as *open* transition systems. The second one, used e.g. in [4], is based on the

[1] Research partially funded by EU Integrated Project Sensoria, contract n. 016004.

[2] Email: bravetti@cs.unibo.it

This is a preliminary version. The final version will be electronically published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.com/locate/entcs

assumption to have a process algebraic representation of the whole system, and uses *unlabeled* reduction transitions to represent the system behavior, i.e. no behaviors possibly induced by a context are considered. In the following we will refer to such unlabeled reduction-based transition systems as *closed* transition systems. Note that, sometimes, reduction transitions can also be labeled: such labels however are not used to represent possible interactions with contexts, but are just informative labels describing what is happening internally in the system (they are useful to analyse the system behaviour by, e.g., model checking).

The distinction between open and closed transition systems is important, in particular, in the case we want to express quantitative behaviours as, e.g., prioritized and probabilistic choices. In the closed transition system of a whole system representation only prioritized behaviours (reductions) are represented and probabilistic choices are just internal: a possible representation is to make use of reduction transitions that lead to probability distribution over states (instead of just single states). On the contrary, in open transition systems, we have the problem of explicitely representing priorities and external probabilistic choices: *absolute* quantitative information, such as *priority levels* and *probabilistic weights*, associated to actions whose execution is just "potential", i.e. depends on the behavior of the context. Though very important from an expressive viewpoint, dealing with priorities and external probabilistic choices in open transition systems turned out to be problematic, especially when the issue of defining weak equivalences (that could be congruences) was considered (see, e.g., [2] for priorities): by directly attaching the quantitative information to actions the problem arises on (i) how to compute the quantitative value for synchronized actions and (ii) how to deal with distinguished τ actions carrying different quantitative information in the weak equivalence. A non-compositional way to deal, in a simple way, with the problem of expressing prioritized behaviours and external probabilistic choices in open transition systems is to use *schedulers*: we consider the open transition system of the whole system and we express weights and priority levels to be associated to actions in the scheduler definition. By applying such a scheduler to the (non-quantified) open transition system we obtain a (quantified) closed transition system as described above.

In this paper we propose a compositional solution to the problem above based on the idea of partially closing open transition systems via a process-algebraic operator. More precisely, we consider mixed models with states where the system is still open (towards interaction with a context) and states where the system is already closed. Moreover, we endow actions labeling open transitions with "handles" h: handles are used by the operator to identify the actions to which the quantitative information must be attached. The idea is that (open) parts of a system "P" can be closed via an operator "$P \uparrow G$" that turns τ_h actions whose handle h is specified inside "G" into reduction transitions that take *priority* with respect to labeled transitions (and, therefore, states performing such τ_h actions into closed states). Note that, as probably expected, only τ_h actions (e.g. in CCS synchronized actions), whose execution no longer depends on the context, can be closed by the operator.

In this way, we can use the operator "$P \uparrow G$" to express multi-level priorities by subsequent applications of the operator: actions closed by an inner application of the operator turn out to have higher priority with respect to actions closed by

an outer application of the operator. For instance, by using a CCS-like parallel operator "|", $(a_{h_1}.P + b_{h_2}.Q|R)\uparrow\{h_2\}\uparrow\{h_1\}$, where output actions $\overline{a}$ and $\overline{b}$ occur in R with neutral handle $*$ (so that synchronization in "|", that involves also handles, gives rise to τ_{h_1} and τ_{h_2} actions), represents a prioritized choice between input actions a and b: if R offers synchronization (output) for both of them at the same time then the b action is executed otherwise the synchronization offered by R is executed. Moreover, we can extend the operator "$P \uparrow G$" to also express external probabilistic choices (at some priority level) by assigning weights to handles inside G. For instance, $(a_{h_1}.P+b_{h_2}.Q|R)\uparrow\{\{(h_1,1),(h_2,3)\}\}$, where output actions $\overline{a}$ and $\overline{b}$ occur in R with neutral handle $*$, represents an external probabilistic choice between input actions a and b: if R offers synchronization (output) for both of them at the same time then they are executed with probabilities .25 (a action) and .75 (b action) otherwise the synchronization offered by R is executed with probability 1. Note that, since priority (and closure of external probability in a state) can be actually applied only when the synchronization context is considered and the involved actions turn from potential to internal, the proposed approach, which allows to put the "$P \uparrow G$" operator just outside this context (and not necessarily at the outermost syntactic level) does not "delay" the application of quantitative information with respect to the traditional approach based on attaching quantitative information directly to potential actions. Moreover, the usage of handles allows the closure operator to be applied selectively even to a single choice.

In the context of probabilistic process algebra literature, classifying states into standard states and quantified states is a natural language design choice that is commonly used to expess internal probabilistic choices (see e.g. [1]): this can be easily done by imposing probabilistic reduction transition to take priority with respect to standard action transitions. Moreover, in this respect, the approach that we adopt here gives us the following additional capabilities: (i) by giving the open/closed interpretation to states and by using an operator to both close the open system parts and, at the same time, assign a probabilistic quantification to them, we can additionally express external probability and also multi-level priority just as a consequence of the simple form of priority between the two kind of transitions; (ii) we can use the same technique in the reduced context of pure non-deterministic process algebra to give a solution to the long term open problem of expressing priorities in this context.

In the paper we consider full CCS with recursion: we use operator "$recX.P$" to express guarded and unguarded recursion. We use such a "core" process algebra (where we additionally attach handles to both internal and visible actions) to express open transition systems and we extend it in two subsequent steps: first we just consider non-deterministic prioritized reductions and a simple version of "$P \uparrow G$" where G is just a set of handles (giving us the ability to express multi-level priorities only), then we also endow reductions with target probability distributions (thus expressing non-determinism among probabilistic reductions) and we extend the structure of set G in "$P \uparrow G$" to be composed by set of mappings from handles to weights (where every mapping can generate a probabilistic reduction transition). Note that, since the role of the core process algebra is just to compute τ_h transitions (possibly via process interaction) and "$P \uparrow G$" just acts on such transitions, i.e. we

have a separation in two "layers" of the open transitions and of reduction (closed) transitions where the second ones are prioritized with respect to the first ones, our approach is not bound to the particular choice (CCS) of the core process algebra: we could have used any other process algebra.

Concerning equivalences, we are able to define weak equivalences that are compatible with the proposed technique by considering reduction transitions as the only unobservable τ transitions: the idea is that transitions of open states, even if τ_h, are still incomplete because they are not closed, i.e. we still have to apply quantification to them. More precisely, for both extensions of CCS we consider two kinds of weak equivalences that both deal with open transitions according to standard bisimulation and are distinguished for the treatment of reduction transitions. The first one aggregates reduction transitions in a trace-based manner: when a closed state is reached by an open transition, we just care about which open states are reachable by finite traces of reduction transitions and if non-escable divergence, e.g. a non-escapable loop of reduction transitions, can be reached. The intuition is that, being reduction transitions prioritized, it is natural to assume that a context cannot observe intermediate states in sequences of such transitions. The second one is simply standard observational congruence: Milner's one in the pure non-deterministic case and its probabilistic extension in [8] for transitions leading to probability distributions. Note that, even if obviously the trace-based equivalence aggregates much more than standard (probabilistic) observational congruence, we considered the latter to show that it is possible to make it compatible with multi-level priorities and external/internal probabilities.

As a main result we have that the trace-based equivalence is a congruence for the extension of CCS and that standard observational congruence is a congruence provided that "$\underline{0}$" is interpreted as failure (so that it is allowed to be weakly equivalent to $recX.\tau.X$ without breaking congruence with respect to parallel) and successful termination "$\underline{1}$" is introduced in the process algebra.

We also observe that the aggregated transition system obtained by applying the trace-based equivalence to systems can be derived directly in operational semantics. By using an "aggregating" version of the operational semanitcs, we do not need to apply equivalence to reduce states, but the system state space is reduced directly by the operational semantics while we go from inner syntactic levels to outer ones and the system is progressively closed.

Finally, we build on the non-deterministic/probabilistic algebra by considering: discrete time, where reduction transitions take one time unit before reaching their probabilistic target, and continuous time, where reduction transitions take, instead, a probabilistic duration (denoted by the rate of an exponential distribution), to be executed. In both timed contexts we show that, by considering the trace-based equivalence, we can aggregate more with respect to the traditional lumping-based equivalences over Discrete Time or Continuous Time Markov Chains that correspond to a bisimulation-based matching of reductions. In particular, in the case of continuous time, if the semantics of parallel of reductions just gives rise to their non-deterministic interleaving (i.e. such a semantics it is not modified with respect to the untimed version in order to account for contemporaneous passage of time in reduction transitions) and just steady state probabilities are to be preserved by

equivalence, then thanks to the *insensitivity property*, the trace-based equivalence just reduces to checking the mean overall duration of traces and, like in the untimed case, probabilities to reach non-reducible or divergent states.

The paper is structured as follows. Sect. 2, concerning management of multi-level Priorities in purely non-deterministic process algebra, presents the process algebra for non-deterministic open/closed systems and the related machinery: transition systems, the equivalences, syntax and semantics, congruence results and the aggregated semantics. Sect. 3 extends all the machinery of Sect. 2 to also deal with internal/external probabilistic choices. Finally, Sect. 4 concerns the usage of the closure operator to express discrete and continuous (exponential) time.

2 Multi-level Priorities

2.1 Partially open and partially closed non-deterministic transition systems

Definition 2.1 A non-deterministic open/closed transition system is a quadruple $(S, Lab, \longrightarrow_c, \longrightarrow_o)$, where

- S is a countable set of states,
- Lab is a countable set of labels of open transitions,
- $\longrightarrow_c \subseteq S \times S$ is a transition relation over states of S that represents *closed transitions*, i.e. reduction transitions,
- $\longrightarrow_o \subseteq S \times Lab \times S$ is a transition relation over states of S labeled over Lab that represents *open transitions*,

such that, for any $s \in S$, it holds that: $s \longrightarrow_c$ implies $\nexists l \in Lab : s \xrightarrow{l}_o$.

Note that, in the definition above and in the rest of the paper, we use: $s \xrightarrow{l}_o s'$ to stand for $(s, l, s') \in \longrightarrow_o$ and $s \xrightarrow{l}_o$ to stand for $\exists s' : s \xrightarrow{l}_o s'$ A similar notation is used for (unlabeled) reduction transition relation $\longrightarrow_c$. We assume predicate $\gg$ to single out reducible states, i.e. $s \gg$ if $s \longrightarrow_c$, $s \not\gg$ otherwise.

The constraint in Def. 2.1 guarantees that states of non-deterministic open/closed transition systems that have outgoing closed transitions (reducible states) cannot have outgoing open transitions and vice-versa. As a consequence system states can be classified into *closed system states* (states with outgoing closed transitions) and *open system states* (all other states). In Sect. 2.4 we will see an alternative approach where states with no outgoing transitions are assumed to be closed.

We use $\longrightarrow_c^+$ to denote the transitive closure of $\longrightarrow_c$ and $\longrightarrow_c^*$ to denote the transitive and reflexive closure of $\longrightarrow_c$. Predicate $\uparrow$ singles out (non-escapable) divergent states, i.e. $s \uparrow$ iff $\nexists s' : s \longrightarrow_c^* s' \wedge s' \not\gg$. Note that $s \uparrow$ implies $s \gg$. We assume predicate on states "$s\,(\not\gg \vee \uparrow)$" to be defined as "$(s \not\gg) \vee (s \uparrow)$".

Definition 2.2 A symmetric relation β over non-reducible states [3] of a non-deterministic open/closed transition system $(S, Lab, \longrightarrow_c, \longrightarrow_o)$ is a weak equiv-

[3] In general it is possible to consider also reducible states in the definition of weak equivalences, however this is not needed for defining weak congruences. See [3] for the extended definition.

alence if, whenever $(s_1, s_2) \in \beta$:

- If $s_1 \xrightarrow{l}_o \longrightarrow_c^* s_1' \,(\not\gg \vee \uparrow)$ then, for some s_2', with $s_2 \xrightarrow{l}_o \longrightarrow_c^* s_2' \,(\not\gg \vee \uparrow)$, we have: either $s_1' \uparrow$ and $s_2' \uparrow$, or $(s_1', s_2') \in \beta$.

Two non-reducible states s_1, s_2 are weakly equivalent, written $s_1 \approx s_2$, iff (s_1, s_2) is included in some weak equivalence. ■

Definition 2.3 Two states s_1, s_2 of a non-deterministic open/closed transition system $(S, Lab, \longrightarrow_c, \longrightarrow_o)$ are weakly congruent, written $s_1 \simeq s_2$, iff:

- If $s_1 \xrightarrow{l}_o \longrightarrow_c^* s_1' \,(\not\gg \vee \uparrow)$ then, for some s_2', with $s_2 \xrightarrow{l}_o \longrightarrow_c^* s_2' \,(\not\gg \vee \uparrow)$, we have: either $s_1' \uparrow$ and $s_2' \uparrow$, or $s_1' \approx s_2'$.
- if $s_1 \longrightarrow_c^+ s_1' \,(\not\gg \vee \uparrow)$ then, for some s_2', with $s_2 \longrightarrow_c^+ s_2' \,(\not\gg \vee \uparrow)$, we have: either $s_1' \uparrow$ and $s_2' \uparrow$, or $s_1' \approx s_2'$.

and a symmetrical constraint holds true for moves of s_2 as well. ■

Example 2.4 In the paper we will represent behaviors of states by means of process algebraic terms (as we will detail in the next Sect. 2.2), for the examples below the standard meaning of prefix (where τ represent a reduction transition), recursion and sum can be assumed. $\tau.l.\underline{0} + \tau.recX.\tau.X \not\simeq \tau.l.\underline{0}$ [4] because $\tau.l.\underline{0} + \tau.recX.\tau.X \longrightarrow_c^+ l.\underline{0}$ and it can reach a divergent state, while $\tau.l.\underline{0} \longrightarrow_c^+ l.\underline{0}$ but it cannot reach a divergent state. On the contrary $\tau.l.\underline{0}+\tau.recX.(\tau.l.\underline{0}+\tau.X) \simeq \tau.l.\underline{0}$ because $\tau.l.\underline{0}+\tau.recX.(\tau.l.\underline{0}+\tau.X) \longrightarrow_c^+ l.\underline{0}$ and it cannot reach a divergent state.

2.2 Prioritized process algebra

The set of synchronization names $\mathcal{N}$ is ranged over by $a, b, c, \ldots$. The set of action names $\mathcal{N} \cup \{\overline{a} \mid a \in \mathcal{N}\} \cup \{\tau\}$, which includes input actions, output actions (identified by the overbar) and the special symbol τ to denote synchronized unnamed actions, is denoted by $\mathcal{AN}$, ranged over by $\alpha, \alpha', \ldots$. We extend complementation to the whole $\mathcal{AN}$ by assuming $\overline{\overline{a}} = a$ and $\overline{\tau} = \tau$. The finite set of handle names $\mathcal{H}$ is ranged over by $h, h', \ldots$. We assume synchronizing actions to yield unnamed actions and handlers of synchronizing actions to be composed by a given (arbitrarily defined) binary operator "$\propto$", i.e. when $\overline{a}_{h_1}$ synchronizes with a_{h_2} we get $\tau_{h_1 \propto h_2}$. From a modeling viewpoint it is convenient to adopt an operator "$\propto$" that has a neutral element, i.e. an handle $*$ (called neutral handle) such that $* \propto h = h \propto * = h$ for every handle h. The set of open actions (actions with handle) is denoted by $OAct = \{\alpha_h \mid \alpha \in \mathcal{AN} \wedge h \in \mathcal{H}\}$. The set of (all) actions is denoted by $Act = OAct \cup \{\tau\}$, that includes τ to express closed actions (actions without handle). The set of term variables is Var, ranged over by $X, Y, \ldots$. The set $\mathcal{E}$ of behavior expressions, ranged over by E, F is defined by

$$E ::= \underline{0} \mid X \mid \alpha_h.E \mid \tau.E \mid E + E \mid E|E \mid E\backslash L \mid E[\varphi] \mid E \uparrow G \mid recX.E$$

where L is a finite subset of $\mathcal{N}$, G is a finite subset of $\mathcal{H}$ and φ is a relabeling function over Act such that: (i) For every $\alpha \in \mathcal{AN}, h \in \mathcal{H}$ there exists α' such

[4] We assume syntactical precedence of prefix w.r.t. other operators when writing terms.

that $\varphi(\alpha_h) = \alpha'_h$; (ii) $\varphi(\tau) = \tau$; and (iii) $\varphi(\overline{\alpha}) = \overline{\varphi(\alpha)}$. The meaning of the operators is the standard one of [6,7], where "$recX.E$" denotes recursion. The main differences and novelties are the following ones. Closed actions (actions τ) give rise to reduction (closed) transitions and are assumed to be prioritized with respect to open actions (actions α_h) that give rise to open transitions. The prioritization operator "$E \uparrow G''$ turns unnamed open actions τ_h performable by E whose handle h is in G into closed actions τ thus turning them into prioritized actions and cutting possible unprioritized alternative behaviors. Closed terms are terms that do not include free variables (i.e. variables X not bound by a "$recX.E$" operator) and are called *processes*. The set $\mathcal{P}$ of processes is ranged over by P,Q,R. We omit trailing $\underline{0}$ when writing process terms.

The semantics of processes gives rise to the non-deterministic open/closed transition system $(\mathcal{P}, OAct, \longrightarrow_c, \longrightarrow_o)$, where $\longrightarrow_c$ (here denoted simply by $\longrightarrow$ and by explicit use of τ reduction labels) and $\longrightarrow_o$ (here denoted simply by $\longrightarrow$) are defined via structural operational semantics by the rules in Tables 1 and 2, plus symmetric rules. In Table 1 we take γ to range over the set of all actions Act: in the symmetric communication rule the handle of the τ transition is still $h_1 \propto h_2$, with h_1 handle of the output action and h_2 handle of the input action. $type(\gamma)$ yields the name in $\mathcal{N}$ of the action γ or τ if γ is an unnamed synchronized action (i.e. $\gamma = \tau$ or $\gamma = \tau_h$ for some handle h).

Example 2.5 The (non-deterministic open/closed) transition system of $\tau.P + \alpha_h.Q$ is the same as that of $\tau.P$. The transition system of $\tau_h.P + \alpha_{h'}.Q \uparrow \{h\}$, where $h' \neq h$, is the same as that of $\tau.P$.

The transition system of $(a_{h_1}.P + b_{h_2}.Q|R) \uparrow \{h_2\} \uparrow \{h_1\}$, where output actions $\overline{a}$ and $\overline{b}$ occurr in R with neutral handle $*$, represents a prioritized choice between input actions a and b: if R offers synchronization (output) for both of them at the same time then the b action is executed (since "$\uparrow \{h_2\}$" syntactically occurs before "$\uparrow \{h_1\}$") otherwise the synchronization offered by R is executed. For instance, if R is $\overline{a}_*.P' + \overline{b}_*.Q'$ then the transition system of the whole system is the same as that of $\tau.(Q|Q')$. If R is $\overline{a}_*.P'$ then the transition system of the whole system is the same as that of $\tau.(P|P')$. If R is $\overline{b}_*.P'$ then the transition system of the whole system is the same as that of $\tau.(Q|Q')$. The transition system of $(P|Q|\overline{a}_*) \uparrow \{h_2\} \uparrow \{h_1\}$, where input action a occurrs in P with handle h_1 and in Q with handle h_2, represents a prioritized choice between the two input actions a: if both P and Q offer synchronization (input) on a at the same time then the a action of Q is executed (since "$\uparrow \{h_2\}$" syntactically occurs before "$\uparrow \{h_1\}$") otherwise the synchronization offered by either P or Q is executed.

In general we can express multilevel priority by using operator $P \uparrow G$ to successively prioritize (and close) actions. We can use

$$P \uparrow G_n \uparrow G_{n-1} \ldots \uparrow G_1$$

to express that actions whose handle (after synchronization) belongs to G_n are at priority level n, actions whose handle belongs to G_{n-1} are at a lower priority level $n-1$, and so on...: actions whose handle belongs to G_1 are at the lowest (supposing that all actions used in P have been closed/prioritized) priority level 1.

$$\gamma.P \xrightarrow{\gamma} P$$

$$\frac{P \xrightarrow{\alpha_h} P' \quad Q \not\gg}{P + Q \xrightarrow{\alpha_h} P'} \qquad \frac{P \xrightarrow{\alpha_h} P' \quad Q \not\gg}{P|Q \xrightarrow{\alpha_h} P'|Q}$$

$$\frac{P \xrightarrow{\overline{a}_{h_1}} P' \quad Q \xrightarrow{a_{h_2}} Q'}{P|Q \xrightarrow{\tau_{h_1 \propto h_2}} P'|Q'}$$

$$\frac{P \xrightarrow{\gamma} P'}{P\backslash L \xrightarrow{\gamma} P'\backslash L} \; type(\gamma) \notin L \qquad \frac{P \xrightarrow{\gamma} P'}{P[\varphi] \xrightarrow{\varphi(\gamma)} P'[\varphi]}$$

$$\frac{P \xrightarrow{\gamma} P' \quad \nexists h \in G : P \xrightarrow{\tau_h}}{P\uparrow G \xrightarrow{\gamma} P'\uparrow G}$$

$$\frac{P\{recX.P/X\} \xrightarrow{\gamma} P'}{recX.P \xrightarrow{\gamma} P'}$$

Table 1
Proposed variant of standard structural operational rules

$$\frac{P \xrightarrow{\tau} P'}{P + Q \xrightarrow{\tau} P'} \qquad \frac{P \xrightarrow{\tau} P'}{P|Q \xrightarrow{\tau} P'|Q}$$

$$\frac{P \xrightarrow{\tau_h} P'}{P\uparrow G \xrightarrow{\tau} P'\uparrow G} \; h \in G$$

Table 2
Additional rules for non-deterministic reduction transitions

Note that (i) closing/prioritizing actions makes it possible to to be closed (ii) closing/prioritizing actions does not necessarily happen at the outermost syntactic level, like in the scenario above, where a similar effect could be obtained by just applying external (prioritized) schedulers to the transition system of P: synchronized actions should be closed at the innermost possible syntactic level so to make effective compositional reduction by means of the weak equivalence.

Theorem 2.6 *Weak congruence "$\simeq$" is a congruence with respect to all the operators of the prioritized process algebra.*

Proof. We first show that weak equivalence $\approx$ is a congruence for non-reducible processes with respect to static operators. We start with the parallel operator "$|$". It is sufficient to show that:

$$\beta = \{(P_1|Q, P_2|Q) \mid P_1 \not\gg \wedge P_2 \not\gg \wedge Q \not\gg \wedge P_1 \approx P_2\}$$

is a weak bisimulation. Supposed that $P_1|Q \xrightarrow{\alpha_h} P_1'|Q' \longrightarrow^* P_1''|Q''(\not\gg \vee \uparrow)$ we have, due to the simple interleaving semantics of reduction transitions in "$|$":

$P_1' \longrightarrow^* P_1''$ and $Q' \longrightarrow^* Q''$. We additionally consider $P_1'''(\not\gg \vee \uparrow)$ such that $P_1'' \longrightarrow^* P_1'''$.

We have three cases depending on how the $\xrightarrow{\alpha_h}$ is derived:

- $\alpha \neq \tau \wedge P_1 \xrightarrow{\alpha_h} P_1' \wedge Q' = Q$. Since $Q' = Q \not\gg$ we also have $Q'' = Q'$. Since $P_1 \approx P_2$, there exist P_2', P_2'' such that $P_2 \xrightarrow{\alpha_h} P_2' \longrightarrow^* P_2''(\not\gg \vee \uparrow)$ with either $P_1''' \uparrow \wedge P_2'' \uparrow$ or $P_1''' \not\gg \wedge P_2'' \not\gg$ and $P_1''' \approx P_2''$.
  Therefore we have: $P_2|Q \xrightarrow{\alpha_h} P_2'|Q \longrightarrow^* P_2''|Q$. If we now use $Q'' = Q \not\gg$ and $P_1''' \uparrow \iff P_1'' \uparrow$ and $P_1'' \not\gg \Rightarrow P_1''' = P_1''$, we have $P_2''|Q(\not\gg \vee \uparrow)$ and either $(P_1''|Q \uparrow \wedge P_2''|Q \uparrow)$ or $(P_1''|Q \beta P_2''|Q)$ and we are done.
- $\alpha \neq \tau \wedge Q \xrightarrow{\alpha_h} Q' \wedge P_1' = P_1$. Since $P_1' = P_1 \not\gg$ we also have $P_1'' = P_1'$. We have directly $P_2|Q \xrightarrow{\alpha_h} P_2|Q' \longrightarrow^* P_2|Q''$. If we now use $P_1'' = P_1 \not\gg$, we have $P_2|Q''(\not\gg \vee \uparrow)$ and either $(P_1|Q'' \uparrow \wedge P_2|Q'' \uparrow)$ or $(P_1|Q'' \beta P_2|Q'')$ and we are done.
- $\alpha = \tau \wedge P_1 \xrightarrow{\alpha'_h} P_1' \wedge Q \xrightarrow{\overline{\alpha'}_h} Q'$ with $\alpha' \neq \tau$. Since $P_1 \approx P_2$, there exist P_2', P_2'' such that $P_2 \xrightarrow{\alpha'_h} P_2' \longrightarrow^* P_2''(\not\gg \vee \uparrow)$ with either $P_1''' \uparrow \wedge P_2'' \uparrow$ or $P_1''' \not\gg \wedge P_2'' \not\gg$ and $P_1''' \approx P_2''$.
  Therefore we have: $P_2|Q \xrightarrow{\tau_h} P_2'|Q' \longrightarrow^* P_2''|Q''$. If we now use $P_1''' \uparrow \iff P_1'' \uparrow$ and $P_1'' \not\gg \Rightarrow P_1''' = P_1''$, we have $P_2''|Q''(\not\gg \vee \uparrow)$ and either $(P_1''|Q'' \uparrow \wedge P_2''|Q'' \uparrow)$ or $(P_1''|Q'' \beta P_2''|Q'')$ and we are done.

The proof for the other static operators, i.e. relabeling and restriction is just a much simplified version of the above proof as in the standard case. We instead report the proof of for the new operator "$P \uparrow G$" that is special because reduction transitions can be generated. It is sufficient to show that:

$$\beta = \{(P_1 \uparrow G, P_2 \uparrow G) \mid P_1 \uparrow G \not\gg \wedge P_2 \uparrow G \not\gg \wedge P_1 \approx P_2\}$$

is a weak bisimulation. Let us suppose that $P_1 \uparrow G \xrightarrow{\alpha_h} P_1' \uparrow G \longrightarrow^* P_1'' \uparrow G(\not\gg \vee \uparrow)$. We preliminary consider $P_1'''(\not\gg \vee \uparrow)$ such that $P_1'' \longrightarrow^* P_1'''$. We have that, since reduction transitions executed by the "$\uparrow G$" operator are either copied or generated by closure, there exists $n \geq 0$ and non-reducible processes $P_1^1, \ldots, P_1^n$ such that $P_1' \longrightarrow^* P_1^1$ and, for $1 \leq i \leq n$, $P_1^i \xrightarrow{\tau_{h_i}} \longrightarrow^* P_1^{i+1}$ with $h_i \in G$, where $P_1^{n+1} = P_1'''$. We have that:

- $P_1 \xrightarrow{\alpha_h} P_1'$ and there is no $h' \in G$ such that $P_1 \xrightarrow{\tau_{h'}}$. Therefore there exist P_2', P_2^1 such that $P_2 \xrightarrow{\alpha_h} P_2' \longrightarrow^* P_2^1(\not\gg \vee \uparrow)$ with either $P_1^1 \uparrow \wedge P_2^1 \uparrow$ or $P_1^1 \not\gg \wedge P_2^1 \not\gg$ and $P_1^1 \approx P_2^1$. Note that, if $n \geq 1$ then $P_2^1 \not\gg$. Moreover there is no $h' \in G$ such that $P_2 \xrightarrow{\tau_{h'}}$ (easly proved by contradiction).
- There exist non-reducible processes $P_2^2, \ldots, P_2^n$ such that, for $1 \leq i \leq n-1$, we have $P_2^i \xrightarrow{\tau_{h_i}} \longrightarrow^* P_2^{i+1}$ with $P_1^{i+1} \approx P_2^{i+1}$. Finally, if $n \geq 1$, there exist P_2^{n+1} such that $P_2^n \xrightarrow{\tau_{h_i}} \longrightarrow^* P_2^{n+1}(\not\gg \vee \uparrow)$ with either $P_1^{n+1} \uparrow \wedge P_2^{n+1} \uparrow$ or $P_1^{n+1} \not\gg \wedge P_2^{n+1} \not\gg$ and $P_1^{n+1} \approx P_2^{n+1}$.
- In conclusion, we have $P_2 \uparrow G \xrightarrow{\alpha_h} P_2' \uparrow G \longrightarrow^* P_2^{n+1} \uparrow G$. We observe that

$(P_1^{n+1} \upharpoonright G) \upharpoonright \Rightarrow (P_2^{n+1} \upharpoonright G) \upharpoonright$. This holds because, if P_1^{n+1} cannot reach, via reduction transitions or τ_h transitions with $h \in G$, a non-reducible state that performs no τ_h actions with $h \in G$; then also P_2^{n+1} cannot reach, via the same kind of transitions, such a state (easily proved, by contradiction, by subsequently matching $\xrightarrow{\tau_h} \longrightarrow^*$ transitions with $h \in G$, moving from equivalent states to equivalent states, similarly as done with the i-th indexed sequence above, and showing that P_1''' would reach a non-reducible state that performs no τ_h actions with $h \in G$). We also observe that $(P_1^{n+1} \upharpoonright G)\not\gg \Rightarrow (P_2^{n+1} \upharpoonright G)\not\gg$ and we recall $P_1''' = P_1^{n+1}$.
If we now use such statements and $P_1''\not\gg \Rightarrow P_1''' = P_1''$, we have $P_2^{n+1} \upharpoonright G(\not\gg \vee \upharpoonright)$ and either $(P_1'' \upharpoonright G) \upharpoonright \wedge (P_2^{n+1} \upharpoonright G) \upharpoonright$, or $P_1'' \upharpoonright G \beta P_2^{n+1} \upharpoonright G$ and we are done.

The congruence of weak congruence over all the operators can be showed by just applying the definition of weak congruence and, for static operators, by resorting to congruence of weak bisimulation. The only non-trivial case is parallel that we sketch in the following.
Let us suppose $P_1 \simeq P_2$. Let us consider first $P_1|Q \xrightarrow{\alpha_h} P_1'|Q' \longrightarrow^* P_1''|Q''(\not\gg \vee \upharpoonright)$. We have three cases for the moves of P_1 and Q similar to the ones considered above for the congruence of weak bisimulation. In the case of a move of P_1, a corresponding move of P_2 must exist and either both divergent processes or non-reducible weak equivalent processes P_1'' and P_2'' are reached. In the case they are non-reducible and Q'' is non-reducible as well, due to congruence of weak bisimulation with respect to parallel we obtain $P_1''|Q''$ and $P_2''|Q''$.
Let us now consider $P_1|Q \longrightarrow^+ P_1'|Q'(\not\gg \vee \upharpoonright)$. In the case P_1 is reducible and originates some moves in the reduction sequence, a corresponding move of P_2 must exist and either both divergent processes or non-reducible weak equivalent processes P_1' and P_2' are reached. The proof then concludes as in the previous case. □

2.3 Aggregating directly in operational semantics

The idea is that we can represent the behavior of a system in a minimal aggregated way by just saying which states s are reducible, i.e. such that $s \gg$, and by showing directly (i) which non-reducible states s' are reachable by reducible states s, i.e. $s \longrightarrow_c^+ s' \wedge s'\not\gg$, and (ii) weather a divergence state is reachable by reducible states s, i.e. $s \longrightarrow_c^+ s' \wedge s' \upharpoonright$ for some s'; instead of including all $\longrightarrow_c$ transitions in labeled transition systems. By doing this, we do not need to apply equivalence to reduce states, but the system state space is reduced directly by the operational semantics, while we go from inner syntactic levels to outer ones and the system is progressively closed.

Definition 2.7 A non-deterministic aggregated open/closed transition system is a quintuple $(S, Lab, Red, -\!\!\rightarrow_c, \longrightarrow_o)$, where

- S is a countable set of states,
- Lab is a countable set of labels of open transitions,
- Red is the subset of S of reducible states,
- $-\!\!\rightarrow_c \subseteq Red \times \{(S - Red) \cup \{\upharpoonright\}\}$ is a transition relation, leading directly from

reducible states to non-reducible states or to divergence "↑", that represents multiple *closed transitions*

- $\longrightarrow_o \subseteq (S - Red) \times Lab \times S$ is a transition relation labeled over Lab that represents *open transitions*,

Similarly as before, given such a transition system, we use predicate $\gg$ to single out reducible states, i.e. $s \gg$ if $s \in Red$, $s \not\gg$ otherwise. We use $\hat{s}$ to range over $S \cup \{\uparrow\}$.

The aggregated semantics of processes can be obtained, by determining Red and $--\rightarrow_c$ from $\longrightarrow_c$ as explained above and by just leaving $\longrightarrow_o$ unchanged, from the semantics of Sect. 2.2.

Equivalence over non-deterministic aggregated open/closed transition system can be directly defined (by simply applying the correspondance above) as follows.

Definition 2.8 A symmetric relation β over non-reducible states of a non-deterministic aggregated open/closed transition system $(S, Lab, Red, --\rightarrow_c, \longrightarrow_o)$ is a weak equivalence if, whenever $(s_1, s_2) \in \beta$:

- If $s_1 \xrightarrow{l}_o s_1'$ and $(s_1' --\rightarrow_c \hat{s}_1''$ or $\hat{s}_1'' = s_1' \not\gg)$ then, for some s_2' and $\hat{s}_2''$, with $s_2 \xrightarrow{l}_o s_2'$ and $(s_2' --\rightarrow_c \hat{s}_2''$ or $\hat{s}_2'' = s_2' \not\gg)$, we have either $\hat{s}_1' = \hat{s}_2' = \uparrow$ or $(\hat{s}_1', \hat{s}_2') \in \beta$.

Two non-reducible states s_1, s_2 are weakly equivalent, written $s_1 \approx s_2$, iff (s_1, s_2) is included in some weak equivalence. ■

Definition 2.9 Two states s_1, s_2 of a non-deterministic aggregated open/closed transition system $(S, Lab, Red, --\rightarrow_c, \longrightarrow_o)$ are weakly congruent, written $s_1 \simeq s_2$, iff:

- If $s_1 \xrightarrow{l}_o s_1'$ and $(s_1' --\rightarrow_c \hat{s}_1''$ or $\hat{s}_1'' = s_1' \not\gg)$ then, for some s_2' and $\hat{s}_2''$, with $s_2 \xrightarrow{l}_o s_2'$ and $(s_2' --\rightarrow_c \hat{s}_2''$ or $\hat{s}_2'' = s_2' \not\gg)$, we have either $\hat{s}_1' = \hat{s}_2' = \uparrow$ or $\hat{s}_1' \approx \hat{s}_2'$.
- If $s_1 --\rightarrow_c \hat{s}_1'$ then, for some s_2', with $s_2 --\rightarrow_c \hat{s}_2'$, we have either $\hat{s}_1' = \hat{s}_2' = \uparrow$ or $\hat{s}_1' \approx \hat{s}_2'$.

and a symmetrical constraint holds true for moves of s_2 as well. ■

The aggregated semantics can be also obtained directly from processes as follows. The non-deterministic aggregated open/closed transition system is $(\mathcal{P}, OAct, Red, --\rightarrow_c, \longrightarrow_o)$, where the set of reducible states Red is taken to be the smallest subset of $\mathcal{P}$ that includes terms $\tau.P$ for every $P \in \mathcal{P}$ and is such that

$$\begin{aligned}
&P \in Red && \Longrightarrow P + Q, Q + P, P|Q, Q|P, P\backslash L, P[\varphi], P \uparrow G \in Red \\
&P \xrightarrow{\tau_h}_o \ \wedge \ h \in G && \Longrightarrow P \uparrow G \in Red \\
&P\{recX.P/X\} \in Red && \Longrightarrow recX.P \in Red
\end{aligned}$$

and $\longrightarrow_o$ (here denoted simply by $\longrightarrow$) is still defined by the rules of Table 1 plus symmetric rules; however, differently from Sect. 2.2, here we take γ to just

$$\frac{P \gg \quad \nexists P' : P \dashrightarrow P'}{P \dashrightarrow \uparrow}$$

$$\frac{P \not\gg}{\tau.P \dashrightarrow P} \qquad \frac{P \dashrightarrow \hat{P}'}{\tau.P \dashrightarrow \hat{P}'}$$

$$\frac{P \dashrightarrow \hat{P}'}{P + Q \dashrightarrow \hat{P}'}$$

$$\frac{P \dashrightarrow \hat{P}' \quad Q \not\gg}{P|Q \dashrightarrow \hat{P}'|Q} \qquad \frac{P \dashrightarrow \hat{P}' \quad Q \dashrightarrow \hat{Q}'}{P|Q \dashrightarrow \hat{P}'|\hat{Q}'}$$

$$\frac{P \dashrightarrow \hat{P}'}{P\backslash L \dashrightarrow \hat{P}'\backslash L} \qquad \frac{P \dashrightarrow \hat{P}'}{P[\varphi] \dashrightarrow \hat{P}'[\varphi]}$$

$$\frac{P \xrightarrow{\tau_h} P' \quad P' \uparrow G \not\gg}{P \uparrow G \dashrightarrow P' \uparrow G} \; h \in G \qquad \frac{P \xrightarrow{\tau_h} P' \quad P' \uparrow G \dashrightarrow \hat{P}'' \uparrow G}{P \uparrow G \dashrightarrow \hat{P}'' \uparrow G} \; h \in G$$

$$\frac{P \dashrightarrow \hat{P}'}{P \uparrow G \dashrightarrow \hat{P}' \uparrow G} \qquad \frac{P\{recX.P/X\} \dashrightarrow \hat{P}'}{recX.P \dashrightarrow \hat{P}'}$$

Table 3
Additional rules for aggregated non-deterministic reduction transitions

range over the set of open actions $OAct$ (thus now excluding τ) and we have that predicate $\gg$ (re-defined above) is directly determined from set Red. Finally, $\dashrightarrow_c$ (here denoted simply by $\dashrightarrow$) is defined by the rules of Table 3 plus symmetric rules, starting from Red and $\longrightarrow_o$. In Table 3, given a context for terms P "$con(P)$", we take "$con(\uparrow)$" to just stand for $\uparrow$. For instance, "$\uparrow |Q$" stands for $\uparrow$. Moreover, we take $'' \uparrow | \uparrow''$ to stand for $\uparrow$.

Note that, we need to preliminarily define set Red and to base the definition of "$\dashrightarrow$" on Red because, in order to establish if a term P can be the target of an aggregated transition that does not lead to divergence, we cannot just require that P does not perform any such aggregated transition. This because, if P is, e.g., $recX.\tau.X$ that does not perform any such aggregated transition (just like $\underline{0}$), then the check above does not work. If unguarded recursion is somehow disallowed (in such a way that also cannot be "dinamically" generated by application of $P \uparrow G$), then the preliminary definition of set Red is not necessary and non-reducibility of states can be just determined by absence of $\dashrightarrow$ transitions.

2.4 A variant compatible with standard observational congruence

The machinery for multilevel priorities can be modified to make it compatible with standard Milner's observational congruence. From the one hand we loose the dis-

tinction between reducible and unreducible states (i.e. $recX.\tau.X$ is now equated by weak bisimulation to $\underline{0}$), from the other hand we observe also intermediate (reducible) state in τ paths, so the equivalence becomes sensitive to the branching structure of τ behaviours and the state space reduction by aggregation of τ transitions (and elimination of intermediate states) less effective.

The crucial modification that we have to do in order to make the process algebra of Sect. 2.2 compatible with standard observational congruence concerns the parallel operator. Modifying the behaviour of parallel is necessary because with the definition of Sect. 2.2, e.g., while $a_h.\underline{0}|recX.\tau.X$ has the same transition system of $recX.\tau.X$, $a_h.\underline{0}|\tau.\underline{0}$ has the same transition system of $\tau.a_h.\underline{0}$, hence observational congruence cannot be a congruence. The problem is that, with observational congruence, $\underline{0}$ (that is weakly bisimilar to $recX.\tau.X$) must be considered by the parallel as a failure event that makes the whole system fail: i.e. the parallel must be such that the behaviour of $P|\underline{0}$ is that of $\underline{0}$ for any P.

The wanted behaviour for parallel is obtained as follows. We interpret $\underline{0}$ as failure and we introduce in the syntax of behaviour expressions $\mathcal{E}$ (and of processes $\mathcal{P}$) successful termination $\underline{1}$. Moreover we introduce a special action $\surd$, denoting successful termination, that we add to the set $OAct$ of open actions. The new operational semantics is obtained by modifying the rule for unsynchronized parallel transitions of Table 1 as follows:

$$\frac{P \xrightarrow{\alpha_h} P' \quad Q \xrightarrow{\gamma}}{P|Q \xrightarrow{\alpha_h} P'|Q} \ \gamma \in OAct$$

where now we have $\surd \in OAct$. An analogous modification of the rule for $+$ (that would lead the behaviour of $P + \underline{0}$ to be that of $\underline{0}$) is optional.

Moreover the following two standard rules, concerning generation of "$\surd$" moves, must be added (to Table 1):

$$\underline{1} \xrightarrow{\surd} \underline{0} \qquad\qquad \frac{P \xrightarrow{\surd} P' \quad Q \xrightarrow{\surd} Q'}{P|Q \xrightarrow{\surd} P'|Q'}$$

From the modeling viewpoint the modifications above require successful termination of processes $\underline{1}$ to be explicitly used by modelers: in a parallel a process that internally fails (i.e. becomes $\underline{0}$) immediately makes the whole system fail. For instance in $a_h.\underline{0}|P$ the whole system fails as soon as the a_h action is executed; in $a_h.\underline{1}|P$, instead, the system waits for termination of P after execution of a_h. Finally note that in the scenario $(a_h.\underline{1}|P)\backslash a$ the system waits for P to execute an output on a as desirable from a modeling viewpoint, i.e. the system does not fail immediately because the lefthand process cannot execute actions. This happens because the cause that disallows action execution is external (the restriction) and not internal.

Theorem 2.10 *Milner's observational congruence is a congruence with respect to all the operators of the prioritized process algebra with successful termination.*

Proof. In the proof we denote Milner's observational congruence (where $OAct$ actions, that include "$\surd$", are the observable actions) by "$\simeq$" and Milner's weak

bisimulation by "$\approx$". $OAct$ is ranged over by γ, $Act = OAct \cup \{\tau\}$ is ranged over by θ. We first show that $\approx$ is a congruence with respect to static operators. We start with the parallel operator "$|$". It is sufficient to show that:

$$\beta = \{(P_1|Q, P_2|Q) \mid P_1 \approx P_2\}$$

is a weak bisimulation. Supposed that $P_1|Q \xrightarrow{\theta} P_1'|Q'$, we have six cases depending on how the $\xrightarrow{\theta}$ is derived:

- $\theta = \alpha_h, \alpha \neq \tau, P_1 \xrightarrow{\alpha_h} P_1' \wedge Q' = Q$. We must have $Q \xrightarrow{\gamma}$ for some γ. Since $P_1 \approx P_2$, there exists P_2' such that $P_2 \stackrel{\alpha_h}{\Longrightarrow} P_2'$ and $P_1' \approx P_2'$. Therefore $P_2|Q \stackrel{\alpha_h}{\Longrightarrow} P_2'|Q$ and $(P_1'|Q\beta P_2'|Q)$ and we are done.
- $\theta = \alpha_h, \alpha \neq \tau, Q \xrightarrow{\alpha_h} Q' \wedge P_1' = P_1$. We must have $P_1 \xrightarrow{\gamma}$ for some γ. Since $P_1 \approx P_2$, $P_2 \stackrel{\gamma}{\Longrightarrow}$ for some γ. Therefore $P_2|Q \stackrel{\alpha_h}{\Longrightarrow} P_2|Q'$ and $(P_1|Q'\beta P_2|Q')$ and we are done.
- $\theta = \tau_h, P_1 \xrightarrow{\alpha'_h} P_1' \wedge Q \xrightarrow{\overline{\alpha'}_h} Q'$ with $\alpha' \neq \tau$. Since $P_1 \approx P_2$, there exists P_2' such that $P_2 \stackrel{\alpha'_h}{\Longrightarrow} P_2'$ and $P_1' \approx P_2'$. Therefore $P_2|Q \stackrel{\tau_h}{\Longrightarrow} P_2'|Q'$ and $(P_1'|Q'\beta P_2'|Q')$ and we are done.
- $\theta = \surd, P_1 \xrightarrow{\surd} P_1' \wedge Q \xrightarrow{\surd} Q'$. This case is totally analogous to the previous one.
- $\theta = \tau, P_1 \xrightarrow{\tau} P_1' \wedge Q' = Q$. Since $P_1 \approx P_2$, there exists P_2' such that $P_2 \stackrel{\hat{\tau}}{\Longrightarrow} P_2'$ and $P_1' \approx P_2'$. Therefore $P_2|Q \stackrel{\hat{\tau}}{\Longrightarrow} P_2'|Q$ and $(P_1'|Q\beta P_2'|Q)$ and we are done.
- $\theta = \tau, Q \xrightarrow{\tau} Q' \wedge P_1' = P_1$. We have immediately $P_2|Q \xrightarrow{\tau} P_2|Q'$ and $(P_1|Q'\beta P_2|Q')$ and we are done.

The proof for the other static operators, i.e. relabeling and restriction is just a much simplified version of the above proof as in the standard case. We instead report the proof of for the new operator "$P \uparrow G$" that is special because reduction transitions can be generated. It is sufficient to show that:

$$\beta = \{(P_1 \uparrow G, P_2 \uparrow G) \mid P_1 \approx P_2\}$$

is a weak bisimulation. Supposed that $P_1 \uparrow G \xrightarrow{\theta} P_1' \uparrow G$, we have three cases depending on how the $\xrightarrow{\theta}$ is derived:

- $\theta = \gamma, P_1 \xrightarrow{\gamma} P_1'$. We must have that $\nexists h \in G : P_1 \xrightarrow{\tau_h}$. Since $P_1 \approx P_2$, there exists P_2' such that $P_2 \stackrel{\gamma}{\Longrightarrow} P_2'$ and $P_1' \approx P_2'$. Moreover, called P_2'' the intermediate state of the weak transition above such that $P_2'' \xrightarrow{\gamma}$ it must be that $\nexists h \in G : P_2'' \xrightarrow{\tau_h}$ (by contradiction, if such a transition existed then $P_2 \stackrel{\tau_h}{\Longrightarrow}$, hence $P_1 \stackrel{\tau_h}{\Longrightarrow}$, but P_1 is not allowed to perform τ actions and it does not perform the τ_h action). Therefore $P_2 \uparrow G \stackrel{\gamma}{\Longrightarrow} P_2' \uparrow G$ and $(P_1' \uparrow G\beta P_2' \uparrow G)$ and we are done.
- $\theta = \tau, P_1 \xrightarrow{\tau_h} P_1'$ with $h \in G$. Since $P_1 \approx P_2$, there exists P_2' such that $P_2 \stackrel{\tau_h}{\Longrightarrow} P_2'$ and $P_1' \approx P_2'$. Therefore $P_2 \uparrow G \stackrel{\tau}{\Longrightarrow} P_2' \uparrow G$ and $(P_1' \uparrow G\beta P_2' \uparrow G)$ and we are done.
- $\theta = \tau, P_1 \xrightarrow{\tau} P_1'$ with $h \in G$. Since $P_1 \approx P_2$, there exists P_2' such that

$P_2 \stackrel{\hat{\tau}}{\Longrightarrow} P_2'$ and $P_1' \approx P_2'$. Therefore $P_2 \uparrow G \stackrel{\hat{\tau}}{\Longrightarrow} P_2' \uparrow G$ and $(P_1' \uparrow G \beta P_2' \uparrow G)$ and we are done.

The congruence of observational congruence over all the operators can be showed by just applying the definition of observational congruence and, for static operators, by resorting to congruence of weak bisimulation, as in the standard way. In the case of parallel, supposed $P_1 \simeq P_2$, we consider $P_1|Q \stackrel{\theta}{\longrightarrow} P_1'|Q'$. We have six cases for the moves of P_1 and Q similar to the ones considered above for the congruence of weak bisimulation. □

3 Adding Probabilities

3.1 Partially open and partially closed non-deterministic and probabilistic transition systems

First of all we introduce the following notation that will be used in the rest of the paper. Let f be a partial function from an arbitrary domain $\mathcal{D}$ to real numbers $\mathbf{R}$. Given a subset D of $dom(f)$ and supposed that $\sum_{s \in D} f(s) \in \mathbf{R}$, we use $f(D)$ to denote such a sum.

A partial discrete probability distribution over a countable set of states S is a function $\sigma : S \longrightarrow [0,1]$ such that $\sigma(S) \leq 1$. A discrete probability distribution σ is a partial discrete probability distribution such that $\sigma(S) = 1$. We denote by $PDist_S$ the set of discrete probability distributions over states S.

In the case S is infinite, it is convinient to introduce the following notatation to denote discrete probability distributions in a finite way. Given a partial function f from S to $[0,1]$ such that $\sigma(dom(f)) = 1$, we use it to denote a probability distribution by writing σ_f defined as: $\sigma_f(s) = f(s)$ if $s \in dom(f)$, $\sigma_f(s) = 0$ otherwise.

Definition 3.1 A non-deterministic/probabilistic open/closed transition system is a quadruple $(S, Lab, \longrightarrow_c, \longrightarrow_o)$, where

- S is a countable set of states,
- Lab is a countable set of labels of open transitions,
- $\longrightarrow_c \subseteq S \times PDist_S$ is a transition relation from states of S to discrete probability distributions over S that represents *closed transitions*, i.e. reduction transitions,
- $\longrightarrow_o \subseteq S \times Lab \times S$ is a transition relation over states of S labeled over Lab that represents *open transitions*,

such that, for any $s \in S$, it holds that: $s \longrightarrow_c$ implies $\nexists l \in Lab : s \stackrel{l}{\longrightarrow}_o$.

Note that, in the definition above and in the rest of the paper, we use: $s \longrightarrow_c \sigma$ to stand for $(s, \sigma) \in \longrightarrow_c$ and $s \longrightarrow_c$ to stand for $\exists \sigma : s \longrightarrow_c \sigma$. We assume predicate $\gg$ to single out reducible states, i.e. $s \gg$ if $s \longrightarrow_c$, $s \not\gg$ otherwise.

We extend predicates $P(s)$ defined on states to hold on discrete probability distributions over states as follows: $P(\sigma)$ iff $\forall\, s \in S.\, \sigma(s) > 0 \Rightarrow P(s)$. For instance, $\sigma \not\gg$ stands for $\forall\, s \in S.\, \sigma(s) > 0 \Rightarrow s \not\gg$. Moreover, given a predicate

$P(s)$ defined on states, we take: S_P to denote the subset of S of states s that satisfy $P(s)$, i.e. $S_P = \{s \in S \mid P(s)\}$; σ_P to denote the partial discrete probability distribution obtained from σ by considering only probability associated to states s that satisfy $P(s)$, i.e. $\forall\, s \in S$ we have $\sigma_P(s) = \sigma(s)$ if $P(s)$, $\sigma_P(s) = 0$ otherwise. For instance, $S_{\not\gg}$ denotes the set of non-reducible states and $\sigma_{\not\gg}$ is the partial discrete probability distribution obtained from σ by considering only probability associated to non reducible states.

A finite trace tr of reduction transitions is a function $tr : \{1, \ldots, n\} \longrightarrow S$, for some $n \in \mathbf{N}^+$ (the length of the trace), such that for every $i \in \{1, \ldots, n-1\}$ there exists σ such that $tr(i) \longrightarrow_c \sigma$ and $\sigma(tr(i+i)) > 0$. We denote by Tr the set of such traces and by Tr_s the subset of all traces tr in Tr such that $tr(1) = s$. In the following we will denote the states of a trace tr just as tr_1, ..., tr_n standing for $tr(1)$, ..., $tr(n)$. Moreover, given a trace tr of length n, we use tr_{fin} to denote its final state tr_n (the only state of the trace that can be a non-reducible state) and $tr_{\leq i}$, with $i \leq n$, to denote the trace of length i that is a prefix of tr.

A (hystory dependent) scheduler $sched_s$ from a state s is a partial function $sched_s : Tr_s \,\text{-}\!\!\!\circ\!\!\!\rightarrow PDist_S$ such that $sched(tr) = \sigma$ implies $tr_{fin} \longrightarrow_c \sigma$ and satisfies: $tr \in dom(sched_s)$ implies $tr_{\leq n-1} \in dom(sched_s)$ and $sched_s(tr_{\leq n-1})(tr_n) > 0$, where n is the length of tr. Tr_{sched_s}, representing finite traces that can be scheduled going from s all the way until one of scheduler's halt states, is the subset of all traces tr in Tr_s such that $tr \notin dom(sched_s)$, $tr_{\leq n-1} \in dom(sched_s)$ and $sched_s(tr_{\leq n-1})(tr_n) > 0$, where n is the length of tr. The probability of a trace $tr \in Tr_{sched_s}$ of length n under a scheduler $sched_s$ is defined by $prob_{sched_s}(tr) = \prod_{1 \leq i \leq n-1} sched_s(tr_{\leq i})(tr_{i+i})$. [5] A scheduler $sched_s$ is *terminating* (by means of finite traces) if $\sum_{tr \in Tr_{sched_s}} prob_{sched_s}(tr) = 1$. [6] Terminating schedulers from s are ranged over by $tsched_s$.

We define $s \longrightarrow_c^* \sigma$, with $\sigma \in PDist_S$, to hold if and only if there exists a scheduler $tsched_s$ such that for every $s' \in S$ it holds $\sigma(s') = \sum_{tr \in Tr_{tsched_s} \wedge tr_{fin} = s'} prob_{tsched_s}(tr)$. The definition of $s \longrightarrow_c^+ \sigma$ is the same with the additional constraint of $tsched_s \neq \emptyset$. Predicate $\uparrow$ singles out (non-escapable) divergent states, i.e. $s \uparrow$ iff $\nexists \sigma : s \longrightarrow_c^* \sigma \wedge \sigma(S_{\not\gg}) > 0$. Note that $s \uparrow$ implies $s \gg$.

Given an equivalence relation β over states S, we say that two partial discrete probability distributions σ' and σ'' are equivalent, written $\sigma' \equiv_\beta \sigma''$ if, for every equivalence class $C \in S/\beta$, it holds that $\sum_{s \in C} \sigma'(s) = \sum_{s \in C} \sigma''(s)$.

Definition 3.2 An equivalence relation β over non-reducible states of a non-deterministic/probabilistic open/closed transition system $(S, Lab, \longrightarrow_c, \longrightarrow_o)$ is a weak equivalence if, whenever $(s_1, s_2) \in \beta$:

- If $s_1 \xrightarrow{l}_o \longrightarrow_c^* \sigma(\not\gg \vee \uparrow)$ then, for some σ', $s_2 \xrightarrow{l}_o \longrightarrow_c^* \sigma'(\not\gg \vee \uparrow)$ and $\sigma_{\not\gg} \equiv_\beta \sigma'_{\not\gg}$.

Two non-reducible states s_1, s_2 are weakly equivalent, written $s_1 \approx s_2$, iff (s_1, s_2) is included in some weak equivalence. ■

[5] We assume an empty product to yield 1.

[6] We assume an empty summation to yield 0.

Definition 3.3 Two states s_1, s_2 of a non-deterministic/probabilistic open/closed transition system $(S, Lab, \longrightarrow_c, \longrightarrow_o)$ are weakly congruent, written $s_1 \simeq s_2$, iff:

- If $s_1 \stackrel{l}{\longrightarrow}_o \longrightarrow_c^* \sigma(\not\gg \vee \uparrow)$ then, for some σ', $s_2 \stackrel{l}{\longrightarrow}_o \longrightarrow_c^* \sigma'(\not\gg \vee \uparrow)$ and $\sigma_{\not\gg} \equiv_{\approx} \sigma'_{\not\gg}$.
- If $s_1 \longrightarrow_c^+ \sigma(\not\gg \vee \uparrow)$ then, for some σ', $s_2 \longrightarrow_c^+ \sigma'(\not\gg \vee \uparrow)$ and $\sigma_{\not\gg} \equiv_{\approx} \sigma'_{\not\gg}$.

and a symmetrical constraint holds true for moves of s_2 as well. ■

Example 3.4 Below we represent a reduction transition that leads to a probability distribution over states by means of a sum "$[p_1]P_1 + \ldots + [p_n]P_n$" ($\sum_{1 \leq i \leq n} p_i = 1$) [7] where each target state is prefixed by a probability. On the contrary non-deterministic choices between (open or reduction) transitions are still represented via standard "$P + Q$" sums (the formal definitions will be given in next Sect. 3.2). $[.3]l.\underline{0} + [.7]recX.[1]X \not\simeq [1]l.\underline{0}$ because the only distributions σ such that $\sigma(\not\gg \vee \uparrow)$ reachable by $[.3]l.\underline{0} + [.7]recX.[1]X$ assign probability .3 to $l.\underline{0}$ and probability .7 to a divergent state, while the only distribution σ such that $\sigma(\not\gg \vee \uparrow)$ reachable by $[1]l.\underline{0}$ assigns probability 1 to $l.\underline{0}$ (i.e. 0 probability is assigned to divergent states). On the contrary, $[.3]l.\underline{0} + [.7]recX.([1]l.\underline{0} + [1]X) \simeq [1]l.\underline{0}$ (where the choice inside recursion is non-deterministic) because the only distributions σ such that $\sigma(\not\gg \vee \uparrow)$ reachable by $[.3]l.\underline{0} + [.7]recX.([1]l.\underline{0} + [1]X)$ assign probability 1 to $l.\underline{0}$: no divergent states can be reached by the initial state.

3.2 Probabilistic prioritized process algebra

The set $\mathcal{E}$ of behavior expressions, ranged over by E, F is defined by

$$E ::= \underline{0} \mid X \mid \alpha_h.E \mid \sum_{i \in I} [p_i].E_i \mid E + E \mid E|E \mid E\backslash L \mid E[\varphi] \mid E \uparrow G \mid recX.E$$

where $\sum_{i \in I} p_i = 1$, L is a finite subset of $\mathcal{N}$, G is a finite set of partial functions from $\mathcal{H}$ to $\mathbb{R}^+$ (representing weights) whose domains are disjoint and φ is a relabeling function over $OAct$ such that: (i) For every $\alpha \in \mathcal{AN}, h \in \mathcal{H}$ there exists α' such that $\varphi(\alpha_h) = \alpha'_h$; (ii) $\varphi(\overline{\alpha}) = \overline{\varphi(\alpha)}$. $\sum_{i \in I} [p_i].E_i$ represents a (discrete) probabilistic choice among terms E_i, where E_i is chosen with probability p_i. The prioritization operator "$E \uparrow G$", for every partial function $g \in G$, turns all open transitions τ_h performable by E whose handlers h are (distinguished and) in the domain of g, into a single closed reduction transition leading to a probability distribution over the target states of the open tranitions, where probabilities are proportional to the weights associated to the handlers by g. Moreover, as in the pure nondeterministic case, it cuts possible unprioritized alternative open behaviors. Again we assume the set $\mathcal{P}$ of processes (i.e. closed terms) to be ranged over by P,Q.

The semantics of processes gives rise to the non-deterministic/probabilistic open/closed transition system $(\mathcal{P}, OAct, \longrightarrow_c, \longrightarrow_o)$, where $\longrightarrow_c$ (here denoted simply by $\longrightarrow$ with no label) and $\longrightarrow_o$ (here denoted simply by $\longrightarrow$) are defined via structural operational semantics by the rules in Tables 1 and 4, plus

[7] In the case of a distribution where all probability is given to a single target P the sum reduces to $[1]P$.

symmetric rules. In Table 1, differently from Sect. 2.2, here we take γ to just range over the set of open actions $OAct$ (thus now excluding τ that is not considered in this section), and we consider $h \in G$ to be an abuse of notation for $h \in dom(g)$ for some $g \in G$, i.e. $h \in \bigcup_{g \in G} dom(g)$. In Table 4, given a context for terms P "$con(P)$" and a probability distribution σ, we take "$con(\sigma)$" to stand for the probability distribution such that: $con(\sigma)(con(P)) = \sigma(P)$, for every $P \in \mathcal{P}$; $con(\sigma)(P') = 0$, for every $P' \in \mathcal{P}$ that is not in the form $con(P)$ for some P. For instance, $\sigma|Q(P|Q) = \sigma(P)$, for every $P \in \mathcal{P}$; $\sigma|Q(P') = 0$ if P' is not in the form $P|Q$ for some P.

Example 3.5 The (non-deterministic/probabilistic open/closed) transition system of $\sum_{i \in I}[p_i].P + \alpha_h.Q$ is the same as that of $\sum_{i \in I}[p_i].P$. The transition system of $\tau_{h_1}.P_1 + \tau_{h_2}.P_2 + \alpha_{h'}.Q \uparrow \{\{(h_1,1),(h_2,3),(h_3,2)\}\}$, where h_1, h_2, h_3, h' are distinguished handlers, is the same as that of $[.25]P_1 + [.75]P_2$. The transition system of $\tau_{h_1}.P_1 + \tau_{h_2}.P_2 + \tau_{h_3}.P_3 + \tau_{h_4}.P_4 \uparrow \{\{(h_1,1),(h_2,3)\}\{(h_3,1),(h_4,1)\}\{(h',1)\}\}$, where h_1, h_2, h_3, h_4, h' are distinguished handlers, is the same as that of $([.25]P_1 + [.75]P_2) + ([.5]P_3 + [.5]P_4)$. The transition system of $\tau_{h_1}.P_1 + \tau_{h_2}.P_2 + \tau_{h_2}.P_3 + \alpha_{h'}.Q \uparrow \{\{(h_1,1),(h_2,3)\}\}$, where h_1, h_2, h' are distinguished handlers, is the same as that of $([.25]P_1 + [.75]P_2) + ([.25]P_1 + [.75]P_3)$.

The transition system of $(a_{h_1}.P + b_{h_2}.Q|R) \uparrow \{\{(h_1,1),(h_2,3)\}\}$, where output actions $\overline{a}$ and $\overline{b}$ occurr in R with neutral handle $*$, represents an external probabilistic choice between input actions a and b: if R offers synchronization (output) for both of them at the same time then they are executed with probabilities .25 (a action) and .75 (b action) otherwise the synchronization offered by R is executed. The transition system of $(a_{h_1}.P_1 + b_{h_2}.P_2 + c_{h_3}.P_3|R) \uparrow \{\{(h_3,1)\}\} \uparrow \{\{(h_1,1),(h_2,3)\}\}$, where output actions $\overline{a}$, $\overline{b}$ and $\overline{c}$ occurr in R with neutral handle $*$, represents a probabilistic/prioritized choice among input actions a, b and c: if R offers synchronization (output) for all of them at the same time (in general if the synchronization set offered by R includes output c) then the c action is executed (since "$\uparrow \{\{(h_3,1)\}\}$" syntactically occurs before "$\uparrow \{\{(h_1,1),(h_2,3)\}\}$"); otherwise if output on c is not offered and both output on actions a and b are offered then a is executed with probability .25 and b with probability .75; finally if just output on action a or on action b is offered that the correspondig action is executed with probability 1.

In general we can express (external) probabilistic choices at multiple priority levels by using $P \uparrow G$ to successively prioritize (and close) actions. We can use

$$P \uparrow G_n \uparrow G_{n-1} \ldots \uparrow G_1$$

to express that actions whose handle (after synchronization) belongs to G_n are at priority level n and a non-deterministic/probabilistic choice among them occurs based on the weight functions in G_n, actions whose handle belongs to G_{n-1} are at a lower priority level $n-1$ and a non-deterministic/probabilistic choice among them occurs based on the weight functions in G_{n-1}, and so on...: actions whose handle belongs to G_1 are at the lowest (supposing that all actions used in P have been closed/prioritized) priority level 1 and a non-deterministic/probabilistic choice among them occurs based on the weight functions in G_1.

As far as the congruence property of "$\simeq$" is concerned, first of all we have to make the definition of $\longrightarrow^*$ and $\longrightarrow^+$ slightly more complicate by using

$$\sum_{i\in I}[p_i].P_i \longrightarrow \sigma_{\{(P_i,\sum_{j\in I:P_j=P_i} p_j)|i\in I\}}$$

$$\frac{P \longrightarrow \sigma}{P+Q \longrightarrow \sigma} \qquad \frac{P \longrightarrow \sigma}{P|Q \longrightarrow \sigma|Q}$$

$$\frac{P \longrightarrow \sigma}{P\backslash L \longrightarrow \sigma\backslash L} \qquad \frac{P \longrightarrow \sigma}{P[\varphi] \longrightarrow \sigma[\varphi]}$$

$$\frac{dom(g)\cap\{h|P \xrightarrow{\tau_h}\} = D \neq \emptyset \quad \forall h\in D.\, P \xrightarrow{\tau_h} P_h}{P\uparrow G \longrightarrow \sigma_{\{(P_h,(\sum_{h'\in D:P_{h'}=P_h} g(h'))/g(D))|h\in D\}}\uparrow G} \; g\in G$$

$$\frac{P \longrightarrow \sigma}{P\uparrow G \longrightarrow \sigma\uparrow G} \qquad \frac{P\{recX.P/X\} \longrightarrow \sigma}{recX.P \longrightarrow \sigma}$$

Table 4
Additional rules for non-deterministic/probabilistic reduction transitions

probabilistic schedulers like in [8]. Such schedulers lead to an increased capability of equating states (without modifying the definition of equivalence): e.g. single system transitions can be matched even if the distribution of one of them is just obtained as a probabilistic combination of the distributions of the others (instead of matching transitions by requiring them to have the same distribution). The adoption of probabilstic schedulers is essential for the aggregation of multiple occurrences of the same states in a probabilistic choice, as e.g. in $[.2]P + [.8]P$ that has the same semantics as $[1]P$, (and ultimately for the aggregation of states belonging to the same equivalence class) to be compatible with equivalence (weak congruence). Moreover, the congruence for the parallel operator is crucially based on the adoption of schedulers with partial visibility. The definition of $\longrightarrow^*$ and $\longrightarrow^+$ must be further complicated by additionally requiring that the corresponding scheduler satisfies the following *partial visibility condition*: the decision about the probabilistic reduction of a given (sequential) process to be performed in a state must depend only on the state of such a process and on the hystory of the states of such a process. In general, when such a scheduler reaches a state: first decides which (sequential) process must perform a probabilistic reduction (this decision can depend on the whole state and on the history of whole states like for schedulers defined in Sect. 3.1), then decides which probabilistic reduction of the chosen process is to be performed by using partial visibility as explained above. Such a property is natural, since, like for probabilities, the decisions about the choice of the reductions to be performed on a process should not depend on the decisions about the choice of the reductions to be performed in the other processes. See [3] for details about congruence.

3.3 Aggregating directly in operational semantics

The idea is that, similarly as in the purely non-deterministic case, we can represent the behavior of a system in a minimal aggregated way by just saying which states s

are reducible, i.e. such that $s \gg$, and by showing directly which distributions $\hat{\sigma}$ over non-reducible states and divergence (denoted by $\uparrow$) are reachable by reducible states s, i.e. $\hat{\sigma} \in PDist_{S_{\not\gg} \cup \{\uparrow\}}$ such that $s \longrightarrow_c^+ \sigma \wedge \sigma(\not\gg \vee \uparrow)$ and $\sigma_{\not\gg} = \hat{\sigma}_{\not\gg}$ instead of including all $\longrightarrow_c$ transitions in labeled transition systems. Note however, that, in the case probabilistic schedulers are adopted, since infinite schedulings are possible, in the general case, we have (continuously) infinite $\hat{\sigma}$ distributions reachable by states. Adopting the non-probabilistic schedulers of Sect. 3.1 does not solve completely the problem, since, e.g., $RecX.[.2]X + [.8]([1]a + [1]b)$ would reach a (enumerable) infinite number of $\hat{\sigma}$ distributions too. Only by restricting to the case where all choices are purely probabilistic (see below), we can be sure of branching finiteness.

Definition 3.6 A non-deterministic/probabilistic aggregated open/closed transition system is a quintuple $(S, Lab, Red, --\!\!\rightarrow_c, \longrightarrow_o)$, where

- S is a countable set of states,
- Lab is a countable set of labels of open transitions,
- Red is the subset of S of reducible states,
- $--\!\!\rightarrow_c \subseteq Red \times PDist_{(S-Red) \cup \{\uparrow\}}$ is a transition relation, leading directly from reducible states to discrete probability distributions over non-reducible states and divergence "$\uparrow$", that represents multiple *closed transitions*
- $\longrightarrow_o \subseteq (S - Red) \times Lab \times S$ is a transition relation labeled over Lab that represents *open transitions*,

As usual, we use predicate $\gg$ to single out reducible states, i.e. $s \gg$ if $s \in Red$, $s \not\gg$ otherwise. We use $\hat{\sigma}$ to range over $PDist_{(S-Red) \cup \{\uparrow\}}$.

The aggregated semantics of processes can be obtained, by determining Red and $--\!\!\rightarrow_c$ from $\longrightarrow_c$ as explained above and by just leaving $\longrightarrow_o$ unchanged, from the semantics of Sect. 2.2.

Equivalence over non-deterministic/probabilistic aggregated open/closed transition system can be directly defined (by simply applying the correspondance above) as follows.

Definition 3.7 A symmetric relation β over non-reducible states of a non-deterministic/probabilistic aggregated open/closed transition system $(S, Lab, Red, --\!\!\rightarrow_c, \longrightarrow_o)$ is a weak equivalence if, whenever $(s_1, s_2) \in \beta$:

- If $s_1 \stackrel{l}{\longrightarrow}_o s_1'$ and $(s_1' --\!\!\rightarrow_c \hat{\sigma}$ or $\hat{\sigma}_{\not\gg}(s_1') = 1)$ then, for some s_2' and $\hat{\sigma}'$, with $s_2 \stackrel{l}{\longrightarrow}_o s_2'$ and $(s_2' --\!\!\rightarrow_c \hat{\sigma}'$ or $\hat{\sigma}'_{\not\gg}(s_2') = 1)$, we have $\hat{\sigma}_{\not\gg} \equiv_\beta \hat{\sigma}'_{\not\gg}$.

Two non-reducible states s_1, s_2 are weakly equivalent, written $s_1 \approx s_2$, iff (s_1, s_2) is included in some weak equivalence. ■

Definition 3.8 Two states s_1, s_2 of a non-deterministic/probabilistic aggregated open/closed transition system $(S, Lab, Red, --\!\!\rightarrow_c, \longrightarrow_o)$ are weakly congruent, written $s_1 \simeq s_2$, iff:

- If $s_1 \stackrel{l}{\longrightarrow}_o s_1'$ and $(s_1' --\!\!\rightarrow_c \hat{\sigma}$ or $\hat{\sigma}_{\not\gg}(s_1') = 1)$ then, for some s_2' and $\hat{\sigma}'$, with

$s_2 \xrightarrow{l}_o s_2'$ and ($s_2' \dashrightarrow_c \hat{\sigma}'$ or $\hat{\sigma}'_{\not\gg}(s_2')=1$), we have $\hat{\sigma}_{\not\gg} \equiv_{\approx} \hat{\sigma}'_{\not\gg}$.

- If $s_1 \dashrightarrow_c \hat{\sigma}$ then, for some $\hat{\sigma}'$, with $s_2 \dashrightarrow_c \hat{\sigma}'$, we have $\hat{\sigma}_{\not\gg} \equiv_{\approx} \hat{\sigma}'_{\not\gg}$.

and a symmetrical constraint holds true for moves of s_2 as well. ■

The aggregated semantics can be also obtained directly from processes similarly as in the non-deterministic case. In the following we show how this can be done in the pure probabilistic case, i.e. for processes such that: (*i*) we have at most one probabilistic choice occurring (unguarded) in the scope of non deterministic choices, (*ii*) for every $P\!\uparrow\! G$ operator, the set G includes a single partial function g. We will then discuss how the presented semantics can be extended to the general non-deterministic/probabilistic case.

The non-deterministic/probabilistic aggregated open/closed transition system is $(\mathcal{P}, OAct, Red, \dashrightarrow_c, \longrightarrow_o)$, where the set of reducible states Red is taken to be the smallest subset of $\mathcal{P}$ that includes terms $\sum_{i\in I}[p_i].P_i$, where P_i are arbitrary processes of $\mathcal{P}$, and is such that

$$\begin{array}{ll} P \in Red & \Longrightarrow P{+}Q, Q{+}P, P|Q, Q|P, P\backslash L, P[\varphi], P\!\uparrow\! G \in Red \\ P \xrightarrow{\tau_h}_o \wedge\ \exists g \in G\colon h \in dom(g) & \Longrightarrow P\!\uparrow\! G \in Red \\ P\{recX.P/X\} \in Red & \Longrightarrow recX.P \in Red \end{array}$$

and $\longrightarrow_o$ (denoted simply by $\longrightarrow$) is still defined by the rules of Table 1 plus symmetric rules; however, differently from Sect. 2.2, here we take γ to just range over the set of open actions $OAct$ (thus now excluding τ) and we have that predicate $\gg$ (re-defined above) is directly determined from set Red. Finally, $\dashrightarrow_c$ (here denoted simply by $\dashrightarrow$) is defined by

$$P \dashrightarrow \hat{\sigma} \quad \Leftrightarrow \quad \forall \hat{P}' \in \mathcal{P} \cup \{\uparrow\}.\, \hat{\sigma}(\hat{P}') = \textstyle\sum_{P \xrightarrow{p} \hat{P}'} p$$

where [8] the probability labeled multi-transition relation $\dashrightarrow$, a multi-set over $\mathcal{P} \times [0,1] \times \mathcal{P}$, is defined by the rules of Table 5 plus symmetric rules, starting from Red and $\longrightarrow$: in Table 5 a transition is taken with multiplicity n if it can be derived in n different ways.

As in the pure non-deterministic case, if unguarged recursion is somehow disallowed, then the preliminary definition of set Red is not necessary and non-reducibility of states can be just determined by absence of $\dashrightarrow$ transitions.

The semantics above can be extended to deal with the general non-deterministic/probabilistic case by just adding information, representing scheduling choices, to reduction transitions. This must be done so to distinguish, in a given reducible state, outgoing probabilistic transitions belonging to different schedulers. The information can be produced as an additional label that records application of operators by their derivation rules. Another possibility is to define the semantics directly on reduction transitions $P \dashrightarrow \hat{\sigma}$. It is possible to do this by defining a preorder over partial probability distributions that coincides with point to point $\leq$ on the probability

[8] In the summation, a distinguished istance of p is considered for each multiple instance of that same transition $P \xrightarrow{p} \hat{P}'$.

associated to states and by defining the semantics of a term to be the one with the minimal paritial probability distributions satisfying the operational semantics. The use of such a pre-order can be seen, for instance, in term $recX.([.4]l.\underline{0} + [.6]X)$, whose semantics is evaluated by starting from a partial probability distribution that assigns zero to all states and incrementing such a partial probability distribution by applying the operational rules.

Note that another way, common in the literature (see, e.g., [1]), to force the system to be purely probabilistic is to adopt a different "+" operator, where probabilistic (reduction) transitions do not resolve the choice. More precisely, by using the notation for (non-aggregated) probabilistic transitions used in this paper and by denoting such an operator with "$\sqcap$", the semantics is:

$$\frac{P \longrightarrow \sigma}{P \sqcap Q \longrightarrow \sigma \sqcap Q}$$

and a symmetric rule, i.e. the same rules for reduction transitions that we have for parallel, while the semantics for open transition is the same as that of "+". Aggregated reduction transitions for "$\sqcap$" are determined with the same rules used for parallel in Table 5. The use of "$\sqcap$" instead of "+" and of restricted $P \upharpoonright G$ operators, where the set G includes a single partial function g, guarantees that all reducible states are purely probabilistic in the aggregated model.

3.4 A variant compatible with probabilistic standard observational congruence

The machinery for internal/external probability and multilevel priorities can be modified to make it compatible with (probabilistic) standard Milner's observational congruence. From the one hand we loose the distinction between reducible and unreducible states (i.e. $recX.\tau.X$ is now equated by weak bisimulation to $\underline{0}$), from the other hand we observe also intermediate (reducible) state in τ paths, so the equivalence becomes sensitive to the branching structure of τ behaviours and the state space reduction by aggregation of τ transitions (and elimination of intermediate states) less effective.

More precisely, we consider probabilistic observational congruence and probabilistic weak bisimulation equivalence as defined in [8] for the so-called "simple model": non-deterministic/ probabilistic open/closed transition systems can be seen as a restriction of such a model where: (i) closed reduction transitions correspond to probabilistic τ transitions and (ii) open labeled transitions correspond to probabilistic labeled (non-τ) transitions that lead to a distribution giving probability 1 to a single target state.

As in the pure non-deterministic case, the crucial modification that we have to do in order to make the process algebra of Sect. 3.2 compatible with probabilistic observational congruence concerns the parallel operator. This because, in terms of the probabilistic algebra we have, e.g., that while $a_h.\underline{0}|recX.[1]X$ has the same transition system of $recX.[1]X$, $a_h.\underline{0}|[1]\underline{0}$ has the same transition system of $[1]a_h.\underline{0}$, hence observational congruence cannot be a congruence.

We must therefore consider $\underline{0}$ (that is weakly bisimilar to $recX.[1]X$) as a failure event. As a consequence: we introduce in the syntax of behaviour expressions $\mathcal{E}$

$$\frac{P \gg \quad \nexists p, P' : P \xrightarrow{p} P'}{P \xrightarrow{1} \uparrow}$$

$$\frac{P_j \not\gg}{\sum_{i\in I}[p_i].P_i \xrightarrow{p_j} P_j}\ j \in I \qquad \frac{P_j \xrightarrow{p} \hat{P}'}{\sum_{i\in I}[p_i].P_i \xrightarrow{p_j \cdot p} \hat{P}'}\ j \in I$$

$$\frac{P \xrightarrow{p} \hat{P}'}{P + Q \xrightarrow{p} \hat{P}'}$$

$$\frac{P \xrightarrow{p} \hat{P}' \quad Q \not\gg}{P|Q \xrightarrow{p} \hat{P}'|Q} \qquad \frac{P \xrightarrow{p'} \hat{P}' \quad Q \xrightarrow{p''} \hat{Q}'}{P|Q \xrightarrow{p'\cdot p''} \hat{P}'|\hat{Q}'}$$

$$\frac{P \xrightarrow{p} \hat{P}'}{P\backslash L \xrightarrow{p} \hat{P}'\backslash L} \qquad \frac{P \xrightarrow{p} \hat{P}'}{P[\varphi] \xrightarrow{p} \hat{P}'[\varphi]}$$

$$\frac{dom(g) \cap \{h' | P \xrightarrow{\tau_{h'}}\} = D \quad P \xrightarrow{\tau_h} P' \quad P' \uparrow G \not\gg}{P \uparrow G \xrightarrow{g(h)/g(D)} P' \uparrow G}\ h \in dom(g), g \in G$$

$$\frac{dom(g) \cap \{h' | P \xrightarrow{\tau_{h'}}\} = D \quad P \xrightarrow{\tau_h} P' \quad P' \uparrow G \xrightarrow{p} \hat{P}'' \uparrow G}{P \uparrow G \xrightarrow{(g(h)/g(D))\cdot p} \hat{P}'' \uparrow G}\ h \in dom(g), g \in G$$

$$\frac{P \xrightarrow{p} \hat{P}'}{P \uparrow G \xrightarrow{p} \hat{P}' \uparrow G} \qquad \frac{P\{recX.P/X\} \xrightarrow{p} \hat{P}'}{recX.P \xrightarrow{p} \hat{P}'}$$

Table 5
Additional rules for aggregated non-deterministic/probabilistic reduction transitions

(and of processes $\mathcal{P}$) successful termination $\underline{1}$, we add to the set $OAct$ of open actions a special action $\surd$, denoting successful termination, and we modify the operational semantics of Table 1 exactly as in the pure non-deterministic case.

As far as the congruence property of probabilistic observatonal congruence is concerned, since, according to the definition given in [8], probabilistic weak equivalence matches single probabilistic reductions to weak transitions (instead of "maximal" weak transitions into weak transitions like in the trace-based equivalence), here the adoption of probabilistic schedulers and the requirement about partial visibility of schedulers are not needed.

In the case we consider a generalized definition of probabilistic weak bisimulation where arbitrary weak transitions must be matched by weak transitions then we have to adopt, as for the trace-based equivalence, the probabilistic schedulers of [8] (a phenomenon similar to the sequence of schedulers in the proof of congruence for the trace-based equivalence with respect to the "$P \uparrow G$" operator arises, due to the

decomposition of the weak transitions into single transitions and re-composition in the other term).

4 Possible extensions: discrete and continuous time

A simple techinque, previoulsy used in the literature (e.g. in the context of continuous time, with exponential distributions), to add capability to express time to a process algebra is to attach the timing information to actions when a model is considered to be complete.

By exploiting our approach, it is possible to do this compositionally: when a part of a system is closed via the "$P \uparrow G$" operator, we can put inside set G the timing information to be attached to actions. We can express, e.g.: (exponentially distributed) continuos time by putting rates of exponential distributions instead of weights inside G and by letting the semantics of "$P \uparrow G$" to additionally label (with respect to that considered in Sect. 3.3) reduction transitions with the assigned (overall) rate; discrete time by assuming that the resulting reduction transition take one time unit to be executed (and by preserving the possibility to include weights inside G to express probabilistic choices).

When timing is considered, trace-based equivalence is established by additionally requiring, w.r.t. that considered in the probabilistic case (see Def. 3.2 and Def. 3.3), that the (continuous or discrete) probability distribution of time associated to matching aggregated reduction transitions ($\longrightarrow_c^*$ or $\longrightarrow_c^+$) must be the same. Moreover in the general case (if we do not want equivalent systems to just preserve particular properties, as we will discuss below) it is necessary to require that, not only the mean probability distribution over states reached by aggregated reduction transitions ($\longrightarrow_c^*$ or $\longrightarrow_c^+$) are compared, but also, probability distributions conditioned on the amount of time taken by aggregated reduction transitions (i.e. a probability distribution is matched for every possible, discrete or continuous, time value).

With respect to bisimulation-based (ordinary lumping-based) markovian aggregation, which requires (as for the equivalence considered Sect. 3.4 for probabilistic systems) to preserve the branching structure of reduction transitions, the obtained equivalence is more coarse. For example, with discrete time

$$[p_1][1]a.\underline{0} + [p_2][1]b.\underline{0} = [1]([p_1]a.\underline{0} + [p_2]b.\underline{0})$$

and with continuous exponentially distributed time

$$[\lambda_1][\mu]a.\underline{0} + [\lambda_2][\mu]b.\underline{0} = [\lambda_1 + \lambda_2]([\mu \cdot \frac{\lambda_1}{(\lambda_1 + \lambda_2)}]a.\underline{0} + [\mu \cdot \frac{\lambda_2}{(\lambda_1 + \lambda_2)}]b.\underline{0}).$$ [9]

Such examples show how, by considering coarser equivalences with respect to bisimulation (as trace-based or even testing-based equivalences), we can reduce the number of system states by merging states (that otherwise would not be mergeable, due

[9] In order for the aggregation to take place it is essential that the states reachable after the first exponential phase have all the same total rate, i.e. sum of rates performable exponential delays (μ in the example) , otherwise the second phase, when aggregated, would become hyperexponentially distributed, instead of just exponentially distributed.

to necessity of preserving the branching structure) and still obtain systems with the same transient state (and consequently steady state) behaviors. For instance, in the example above, the states $[\mu]a.\underline{0}$ and $[\mu]b.\underline{0}$ that are not lumpable (cannot be put in the same quivalence class by markovian bisimulation) can, instead, be merged by considering our equivalance: even if the states are not lumpable such aggregation is correct from a stochastic viewpoint. Similarly, in the discrete time case, for the states $[1]a.\underline{0}$ and $[1]b.\underline{0}$.

Finally, we would like to note that, in the continuous time case, if a parallel operator like that of Sect. 3.2 is considered, where, in the case of parallel of closed states, the reduction transition to be executed is just non-deterministically chosen (i.e. time reduction transitions are non-deterministically interleaved by parallel, thus obtaining a sequentilization of their execution time), then it is possible to adopt a very coarse version of the equivalence which just matches the mean time for performing aggregated transitions (instead of matching the time distribution) and the mean probability distribution over states reached by aggregated reduction transitions (instead of probability distributions conditioned on time). Due to the *insensitivity property* of the considered systems (time distributions are never really contemporaneously executed because of the priority of reduction transitions over open transitions and of the way parallel of closed states is defined) such an equivalence can be a congruence and preseves the steady state behavior of systems. The obtained aggregating power is much greater with respect to the general equivalence above. More precisely every system can be turned into an equivalent aggregated one where reducible states directly reach, via exponential rate-labeled reduction transitions, distributions over non-reducible states or non-escapable divergent states: rates are obtained as the inverse of the mean time for performing aggregated transitions and reached distributions are just given by the mean probability distribution reached by aggregated reduction transitions.

References

[1] S. Andova, *"Process Algebra with Probabilistic Choice"*, in Proc. of *Formal Methods for Real-Time and Probabilistic Systems, 5th International AMAST Workshop*, LNCS 1601:111-129, 1999.

[2] R. Cleaveland, G. Luttgen, V. Natarajan, *"Priority in Process Algebras"*, in Handbook of Process Algebra, Chapter 12, pp. 711-765, Elsevier, 2001

[3] M. Bravetti, Expressing Priorities, External Probabilities and Time in Process Algebra via Mixed Open/Closed Systems Technical report UBLCS-2007-18, Department of computer science, University of Bologna, June 2007.

[4] M. Bravetti, R. Gorrieri, R. Lucchi, G. Zavattaro. *"Quantitative Information in the Tuple Space Coordination Model"*, *Theoretical Computer Science*, 346:1, pages 28-57, Elsevier, 2005.

[5] N. Lynch, R. Segala, F. Vaandrager, *"Observing Branching Structure through Probabilistic Contexts"*, To appear in *Siam Journal on Computing*.
Available at `http://theory.lcs.mit.edu/tds/lynch-pubs.html`

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

[7] R. Milner, *"A complete axiomatization for observational congruence of finite-state behaviours"*, in Information and Computation 81:227-247, 1989

[8] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1995.

A Chart Semantics for the Pi-Calculus

Johannes Borgström[1,2]

TU Berlin

Andrew D. Gordon[3] Andrew Phillips[4]

Microsoft Research

Abstract

We present a graphical semantics for the pi-calculus, that is easier to visualize and better suited to expressing causality and temporal properties than conventional relational semantics. A pi-chart is a finite directed acyclic graph recording a computation in the pi-calculus. Each node represents a process, and each edge either represents a computation step, or a message-passing interaction. Pi-charts enjoy a natural pictorial representation, akin to message sequence charts, in which vertical edges represent control flow and horizontal edges represent data flow based on message passing. A pi-chart represents a single computation starting from its top (the nodes with no ancestors) to its bottom (the nodes with no descendants). Unlike conventional reductions or transitions, the edges in a pi-chart induce ancestry and other causal relations on processes. We give both compositional and operational definitions of pi-charts, and illustrate the additional expressivity afforded by the chart semantics via a series of examples.

Keywords: pi-calculus, causality, message sequence charts.

1 Message Sequence Charts as Process Histories

Message sequence charts (MSCs) are a successful graphical notation for describing the history of interactions between system components running in parallel. They are standardized by the ITU in connection with the Specification and Description Language (SDL) [22,21], and are included, as sequence diagrams, in the Unified Modeling Language (UML) [33]. MSCs are widely used to specify the behaviour of systems made up of multiple components; a substantial literature addresses the problems of defining formal semantics for MSCs and deriving implementation code from MSCs used as specifications [27,2].

This paper explores a different direction, the use of MSCs as a formal semantics, in terms of potential execution histories, for known code. We work within a process calculus,

[1] A long version of this paper is available at http://lamp.epfl.ch/~jobo/chartLONG.pdf

[2] Email: jobo@cs.tu-berlin.de

[3] Email: adg@microsoft.com

[4] Email: aphillip@microsoft.com

This is a preliminary version. The final version will be electronically published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.com/locate/entcs

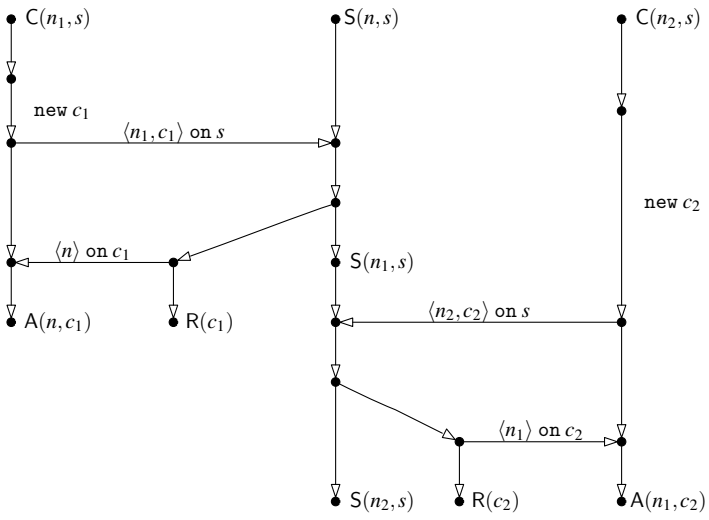
the pi-calculus, although the ideas should apply to other languages. The semantics of the pi-calculus is typically specified as a reaction or reduction relation, or as a labelled transition system [30,39]. We propose a form of MSC as an alternative.

In fact, MSCs are already used informally to illustrate computations in process calculi. For example, Bonelli and Compagnoni [5] visualize intended histories of pi-calculus processes with MSCs. Phillips, Yoshida, and Eisenbach [36] illustrate the semantics of a distributed abstract machine for the boxed ambient calculus [11] with MSCs. Jeffrey and Rathke [23] consider traces induced by a labelled transition system, and make informal connections between these and sequence diagrams. A paper [4] on the TulaFale process language uses an MSC to show an attack on a security protocol. In these papers, the formal semantics is given by relations and MSCs appear only informally. The attraction of MSCs is that they pictorially represent the identity of individual process components as they evolve and interact with other components; the conventional reduction semantics hides this information. Since the history and identity of components is valuable for expressing formal properties of systems, we go further and ask whether MSCs are suitable in themselves as a formal semantics.

To explain some of the basic ideas and to see some of the benefits of a chart semantics for the pi-calculus, we describe a simple example. We suppose there is a single stateful server $\mathsf{S}(n,s)$ which when called with a value n' and a session channel c, responds by sending on c its current state n, provisions a private service $\mathsf{R}(c)$ to handle the session, and changes state to $\mathsf{S}(n',s)$. Here is pi-calculus code for such a server, together with a client $\mathsf{C}(n',s)$ that initiates such a session, and then runs $\mathsf{A}(n,c)$ where n is the previous state of the server and c is the session channel.

$$\mathsf{S}(n,s) := s(n',c).(\overline{c}\langle n\rangle.\mathsf{R}(c) \mid \mathsf{S}(n',s))$$
$$\mathsf{C}(n',s) := (\nu c)\overline{s}\langle n',c\rangle.c(n).\mathsf{A}(n,c)$$

The pi-chart below shows interactions between one server and two clients. Pi-charts are in the spirit of MSCs but do not conform to the letter of the standard [21]. In particular, we allow processes to fork, and to generate fresh names.



A pi-chart is a directed acyclic graph. Both nodes and edges are labelled. As in this example we usually omit some node and edge labels to avoid clutter. Nodes are labelled

with processes. Downward (or oblique) edges represent process evolution and are labelled with *next labels*, including $\texttt{new}\ c$, which represents the generation of a channel c, and ε, which represents the unfolding of a process constant or parallel composition. The next label ε is generally omitted. Horizontal edges represent interaction, and are labelled with *communication labels*, $\langle \widetilde{c} \rangle\ \texttt{on}\ a$.

A pi-chart represents a single computation starting from its top (the nodes with no ancestors) to its bottom (the nodes with no descendants), with restrictions corresponding to any new names. The computation in the example corresponds to the following series of reductions in a conventional reduction semantics.

$$\mathsf{C}(n_1,s) \mid \mathsf{S}(n,s) \mid \mathsf{C}(n_2,s) \rightarrow^* (\nu c_1)(\nu c_2)(\mathsf{A}(n,c_1) \mid \mathsf{R}(c_1) \mid \mathsf{S}(n_2,s) \mid \mathsf{R}(c_2) \mid \mathsf{A}(n_1,c_2))$$

As a means of visualizing computation, pi-charts have advantages over the conventional relational semantics. A series of computation steps in the relational semantics is hard to visualize; listing the series of intermediate states can lead to an overwhelming amount of syntactic detail. Conventionally, reduction and transition relations are closed up to associativity and commutativity of parallel composition. Hence, it is hard to track the evolution of individual threads within a system. One solution is to introduce syntax for abstract locations [13], although this increases the amount of syntactic detail when visualizing reductions. On the other hand, pi-charts have a two dimensional representation that is easily rendered pictorially. The graphical structure allows detail, such as process labels, to be omitted with little risk of ambiguity. Vertical paths in a chart track the evolution of individual processes; in our example, we see that $\mathsf{S}(n,s)$ is an ancestor of $\mathsf{R}(c_1)$, $\mathsf{S}(n_2,s)$, and $\mathsf{R}(c_2)$, but not of the other processes at the bottom of the chart. (There is, though, a causal relation between $\mathsf{S}(n,s)$ and all the processes at the bottom.)

In general, MSCs have been highly successful as a means of visualising and validating dynamic behaviour of concurrent systems, and their graphical representation has also facilitated communication between groups with different backgrounds [28]. We believe that a sequence chart representation of pi-calculus computations could have similar benefits.

We proceed as follows. In Section 2 we formally define a chart semantics for a synchronous pi-calculus with mixed choice. We give three separate inductive characterizations of the set of pi-charts; Theorem 2.4 establishes the equivalence of these characterizations. As evidence for the expressivity of pi-charts, we give a series of examples of correctness properties expressible using charts. Section 3 investigates the relationship of pi-charts to a conventional reduction semantics. Theorem 3.3 shows the relation between the parallel compositions of processes at the top and bottom of a pi-chart coincides with the reflexive and transitive closure of a conventional reduction semantics, up to top-level restrictions. Theorem 3.5 relates structural congruence of processes with a structural congruence on graphs. Sections 4 and 5 conclude and discuss related work.

Appendix A defines the conventional reduction semantics. Appendix B shows how charts can usefully illustrate the behaviour of biological reactions expressed in the pi-calculus. Appendix C is a case study of proving properties expressible with pi-charts. We introduce a type system built from standard notions of name groups, group creation, and usage bounds on channels. Formal data flow and usage properties are conveniently expressed using charts. Theorem C.1 establishes bounds on data flow and channel usage guaranteed by the type system.

2 A Chart Semantics

We consider a polyadic pi-calculus, with synchronous communication, mixed choice, and process constants. Standard variations such as replication operators or asynchronous output can be accommodated in our framework, but we omit the details. The only unusual feature is that we annotate the autonomous τ prefixes with terms t from a free algebra $\mathbf{A}$ over names; these terms serve various purposes, such as representing events (for correspondence assertions [20]) and type annotations (for the system in Appendix C).

Syntax for Pi-Calculus Processes: P

a, c, x	names and variables
$M ::= M + M \mid \overline{a}\langle\widetilde{c}\rangle.P \mid a(\widetilde{x}).P \mid \tau_t.P$	mixed choice
$P, Q, R ::= M \mid (P \mid Q) \mid (\nu a)P \mid \mathsf{A}(\widetilde{c}) \mid \mathbf{0}$	process

Let $\mathbf{P}$ be the set of all processes. Names identify communication channels. We write $\mathrm{fn}(P)$ for the set of names occurring free in P. Let $P\{^{y}/_{x}\}$ be the outcome of substituting y for each free occurrence of x in P. We write $\widetilde{a}$, $\widetilde{c}$, $\widetilde{x}$ for finite tuples of names.

The intended meaning of the process syntax is as follows. An *output* $\overline{a}\langle\widetilde{c}\rangle.P$ sends the tuple $\widetilde{c}$ on channel a, to become P. An *input* $a(\widetilde{x}).P$ receives a tuple $\widetilde{c}$, of the same length as $\widetilde{x}$, off channel a, to become $P\{^{\widetilde{c}}/_{\widetilde{x}}\}$. In $a(\widetilde{x}).P$, the names $\widetilde{x}$ are bound with scope P, and assumed to be pair-wise distinct. A process $\tau_t.P$ autonomously marks the event t, and becomes P. A *choice* $M+N$ behaves either as M or N. A *parallel composition* $P \mid Q$ behaves as P running in parallel with Q. A *restriction* $(\nu a)P$ creates a fresh name a and becomes P; the name a is bound and has scope P. We assume a given *constant library*, a finite collection of process constants, each of which has a *definition*, written $\mathsf{A}(\widetilde{x}) := P$, where $\mathrm{fn}(P) \subseteq \widetilde{x}$. Given such a definition, a process $\mathsf{A}(\widetilde{c})$ behaves as $P\{^{\widetilde{c}}/_{\widetilde{x}}\}$. Finally, $\mathbf{0}$ does nothing.

We identify phrases of syntax up to consistent renaming of bound names; for instance, $(\nu a)P = (\nu b)P\{^{b}/_{a}\}$ if $b \notin \mathrm{fn}(P)$. We also identify processes up to associativity and commutativity of the choice operator.

2.1 Labelled graphs

Charts are particular labelled graphs. Nodes are drawn from an infinite set of *node identifiers*, $\mathbf{I}$, ranged over by ι. Nodes are labelled with pi-calculus processes. Each edge has either a *next label* ($n\ell$) or a *communication label* ($\langle\widetilde{c}\rangle$ `on` a). A next label represents an event, and labels an edge from a process to its successor; the next label `new` x, where the free name x is globally fresh, represents name generation. Annotation t represents a tau step, while next step ε represents all other kinds of process evolution, including unfolding of process constants and parallel compositions. A communication label represents a message passing from an output to an input.

Edge Labels for the Pi-Calculus: nL and L

$n\ell \in \mathbf{nL} ::=$	next label
$\texttt{new}\, x$	name generation
t	annotation
ε	next step
$\ell \in \mathbf{L} ::=$	edge label
$n\ell$	next label
$\langle \widetilde{c} \rangle \texttt{ on } a$	communication

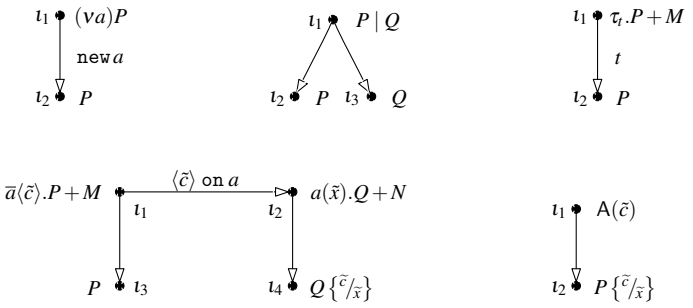
A labelled graph is a pair (N,E) where $\mathsf{N} : \mathbf{I} \to \mathbf{P}$ and $\mathsf{E} : \mathbf{I} \times \mathbf{I} \to \mathbf{L}$ are finite maps. Given $G = (\mathsf{N},\mathsf{E})$, we write N_G for N and E_G for E. A graph G is *well-formed* iff $\mathrm{dom}(\mathsf{E}_G) \subseteq (\mathrm{dom}(\mathsf{N}_G) \times \mathrm{dom}(\mathsf{N}_G))$. The following notations express graphs as compositions of labelled nodes and edges.

$$\begin{aligned} \iota \overset{\ell}{\rightarrowtail} \iota' &:= (\varnothing, \{((\iota,\iota'),\ell)\}) \\ \iota \bullet P &:= (\{(\iota,P)\},\varnothing) \\ G \cup H &:= (\mathsf{N}_G \cup \mathsf{N}_H, \mathsf{E}_G \cup \mathsf{E}_H) \text{ when a well-formed graph} \\ G \setminus H &:= (\mathsf{N}_G \setminus \mathsf{N}_H, \mathsf{E}_G \setminus \mathsf{E}_H) \end{aligned}$$

2.2 *Primitive pi-charts*

We begin our chart semantics by defining a set of primitive pi-charts. Let a *primitive chart* be any instance of one of the following five schemas. Here and elsewhere we omit the ε label from edges. We refer to nodes with the variables $\iota_1, \iota_2, \iota_3, \iota_4$, assumed pair-wise distinct. Let $\mathbf{C}_p$ be the set of primitive charts.

Primitive Charts: $\mathbf{C}_p$



when $\mathsf{A}(\widetilde{x}) := P$

Since we identify processes up to renaming of bound names, from $(\nu a)P$ we get infinitely many primitive charts of the first form above, one for each possible choice of a.

2.3 *The top and bottom of pi-charts*

Each pi-chart has a *top*, the nodes with no predecessors, and a *bottom*, the nodes with no successors. A core idea, formalized later as Theorem 3.3, is that a pi-chart represents a computation starting with the processes at the top, and ending with those at the bottom. We formalize top and bottom below, together with other notations needed for a compositional

definition of pi-charts: $\mathsf{new}(G)$ is the set of names generated within a chart; G_{nil} is the edgeless graph consisting of the terminal nodes of G, that is, those labelled with $\mathbf{0}$.

$$\begin{aligned} G_\top &:= (\{(\iota, P) \mid \mathsf{N}_G(\iota) = P \wedge \neg(\exists \iota', n\ell.\ \mathsf{E}_G(\iota', \iota) = n\ell)\}, \varnothing) \\ G_\bot &:= (\{(\iota, P) \mid \mathsf{N}_G(\iota) = P \wedge \neg(\exists \iota', n\ell.\ \mathsf{E}_G(\iota, \iota') = n\ell)\}, \varnothing) \\ \mathsf{new}(G) &:= \{a \mid \mathtt{new}\ a \in \mathrm{range}(\mathsf{E}_G)\} \\ G_{\mathsf{nil}} &:= (\{(\iota, \mathbf{0}) \mid \mathsf{N}_G(\iota) = \mathbf{0}\}, \varnothing) \\ \mathsf{I}_G &:= \mathrm{dom}(\mathsf{N}_G) \end{aligned}$$

We write $\mathsf{nn}_G(S)$ for $\cup_{\iota \in S} \mathrm{fn}(\mathsf{N}_G(\iota))$ when $S \subseteq \mathsf{I}_G$. When speaking about a particular graph G, we often write $\mathsf{nn}(S)$ for $\mathsf{nn}_G(S)$. We let $\mathsf{nn}(G) := \mathsf{nn}_G(\mathsf{I}_G)$. One invariant we want to preserve is that all names that occur in a chart are either free in the processes at the top of the chart or freshly created. A *well-named* chart is one satisfying $\mathsf{nn}(G) \subseteq \mathsf{nn}(G_\top) \uplus \mathsf{new}(G)$; note that all primitive charts are well-named.

2.4 Three equivalent characterizations of pi-charts

We can now define how to build larger charts from primitive ones. We give three definitions, two compositional and one operational in flavour, and show them equivalent.

Intuitively, two pi-charts may be composed in sequence if the bottom of the first equals the top of the second. Dually, two pi-charts may be composed in parallel if they are completely disjoint. Given these notions, a *pi-chart* is either a singleton chart $\iota \bullet P$, a primitive chart $G \in \mathbf{C}_p$, or a composition $G \cup H$ where G and H are composable, either in sequence or in parallel.

The following definitions make these intuitions precise; various freshness conditions are needed to guarantee global uniqueness of generated names.

Sequential Composition: $S(G,H)$

If G and H are well-formed then $S(G,H)$ iff
(1) $\mathsf{I}_G \cap \mathsf{I}_H = \mathsf{I}_{G_\bot} \setminus \mathsf{I}_{(G_\bot)_{\mathsf{nil}}} = \mathsf{I}_{H_\top}$;
(2) $\mathsf{new}(H) \cap \mathsf{new}(G) = \mathsf{nn}(G_\top) \cap \mathsf{new}(H) = \varnothing$; and
(3) whenever $\iota \in \mathsf{I}_G \cap \mathsf{I}_H$ then $\mathsf{N}_G(\iota) = \mathsf{N}_H(\iota)$.

Parallel Composition: $P(G,H)$

If G and H are well-formed then $P(G,H)$ iff
(1) $\mathsf{I}_G \cap \mathsf{I}_H = \varnothing$; and
(2) $\mathsf{new}(G) \cap \mathsf{new}(H) = \mathsf{nn}(G_\top) \cap \mathsf{new}(H) = \mathsf{nn}(H_\top) \cap \mathsf{new}(G) = \varnothing$.

A First Characterization of Pi-Charts: $\mathbf{C}_{SP}$

$$\frac{}{\iota \bullet P \in \mathbf{C}_{SP}} \qquad \frac{G \in \mathbf{C}_p}{G \in \mathbf{C}_{SP}} \qquad \frac{G,H \in \mathbf{C}_{SP} \quad S(G,H)}{G \cup H \in \mathbf{C}_{SP}} \qquad \frac{G,H \in \mathbf{C}_{SP} \quad P(G,H)}{G \cup H \in \mathbf{C}_{SP}}$$

Although sequential and parallel compositions are intuitive and easy to define, they lack some algebraic properties useful in proofs. As an example, if $P(G_1,G_2)$ and $S(G_1 \cup G_2, H)$, we neither have $S(G_1,H)$ nor $P(G_1,H)$, in general. Moreover, inductive proofs using the

definition of $\mathbf{C}_{SP}$ require two inductive cases, where one ought to suffice. To overcome these problems, we unify parallel and sequential composition into *liberal composition*, and obtain a second definition of pi-charts.

Liberal Composition: $L(G,H)$

If G and H are well-formed then $L(G,H)$ ("G before H") iff
(1) $\mathsf{I}_G \cap \mathsf{I}_H \subseteq \mathsf{I}_{G_\perp}$ and $\mathsf{I}_G \cap \mathsf{I}_H \subseteq \mathsf{I}_{H_\top}$;
(2) $\mathsf{new}(H) \cap \mathsf{new}(G) = \mathsf{nn}(G_\top) \cap \mathsf{new}(H) = \mathsf{nn}(H_\top \setminus G) \cap \mathsf{new}(G) = \varnothing$; and
(3) whenever $\iota \in \mathsf{I}_G \cap \mathsf{I}_H$ then $\mathsf{N}_G(\iota) = \mathsf{N}_H(\iota)$.

A Second Characterization of Pi-Charts: $\mathbf{C}_L$

$$\frac{}{\iota \bullet P \in \mathbf{C}_L} \qquad \frac{G \in \mathbf{C}_p}{G \in \mathbf{C}_L} \qquad \frac{G \in \mathbf{C}_L \quad H \in \mathbf{C}_L \quad L(G,H)}{G \cup H \in \mathbf{C}_L}$$

By comparing definitions, it is clear that liberal composition is more permissive than either parallel or sequential composition. Crucially, liberal composition is associative, and preserves well-namedness.

Lemma 2.1 *Assume that graphs G_1, G_2, G_3 are well-named.*

(1) *If $L(G_1,G_2)$ and $L(G_1 \cup G_2, G_3)$, then $L(G_2,G_3)$ and $L(G_1, G_2 \cup G_3)$.*

(2) *If $L(G_2,G_3)$ and $L(G_1, G_2 \cup G_3)$, then $L(G_1,G_2)$ and $L(G_1 \cup G_2, G_3)$.*

Lemma 2.2 *If G_1, G_2 are well-named and $L(G_1,G_2)$ then $G_1 \cup G_2$ is well-named.*

By associativity (Lemma 2.1) we obtain the following iterative account of $\mathbf{C}_L$.

Lemma 2.3 *$G \in \mathbf{C}_L$ iff there exist pi-charts $H_1, \ldots, H_n \in \mathbf{C}_p \cup \{\iota \bullet P \mid \iota \in \mathbf{I}, P \in \mathbf{P}\}$ such that $G = H_1 \cup \ldots \cup H_n$ and $L(H_1 \cup \cdots \cup H_{i-1}, H_i)$ for each $i \in 2..n$.*

For our final definition, we start with an initial set of unconnected nodes and add primitive charts one by one to the bottom. This amounts to an operational semantics. (We use it as the basis of two separate pi-calculus implementations that output pi-charts in the dot language, suitable for rendering with Graphviz [19].) We define *chart extension* $G \to G'$ ("G extends to G'") as follows, and hence obtain a third characterization of pi-charts.

Chart Extension $G \to G'$ and a Third Characterization of Pi-Charts $\mathbf{C}_I$

$G \to G'$ iff there is $H \in \mathbf{C}_p$ such that $G' = G \cup H$ and $L(G,H)$ and $\mathsf{I}_{H_\top} \subseteq \mathsf{I}_{G_\perp}$
$\mathbf{C}_I := \{G \mid G_\top \to^* G\}$

Theorem 2.4 $\mathbf{C}_{SP} = \mathbf{C}_L = \mathbf{C}_I$

Proof. We begin by proving that $\mathbf{C}_L \subseteq \mathbf{C}_I$, that is, that $G \in \mathbf{C}_L$ implies that $G_\top \to^* G$. Trivially, $L(G_\top, G)$. By Lemma 2.3, there exist primitive pi-charts H_1, ..., H_n such that $G = H_1 \cup \ldots \cup H_n$ and $L(H_1 \cup \cdots \cup H_{i-1}, H_i)$ for each $i \in 1..n$. Since $L(G_\top, H_1 \cup \ldots \cup H_n)$ Lemma 2.1(i) gives that $L(G_\top, H_1 \cup \ldots \cup H_{n-1})$ and $L(G_\top \cup H_1 \cup \ldots \cup H_{n-1}, H_n)$. We also have $\mathsf{I}_{H_{n\top}} \subseteq \mathsf{I}_{(G_\top \cup H_1 \cup \ldots \cup H_{n-1})_\perp}$, so $G_\top \cup H_1 \cup \ldots \cup H_{n-1} \to G$. Inductively, $G_\top \to^n G$.

Secondly, we prove that $\mathbf{C}_I \subseteq \mathbf{C}_{SP}$, that is, that if $G_\top \to^* G$ then $G \in \mathbf{C}_{SP}$, by induction on the number of extensions. For the base case, $G = G_\top = (\mathsf{N}_G, \varnothing) \in \mathbf{C}_{SP}$, by parallel composition of charts of the form $\iota \bullet P$. For the induction case we have $G = G' \cup H$ with

$G' \in \mathbf{C}_I$, $H \in \mathbf{C}_p$, $L(G',H)$ and $\mathsf{I}_{H_\top} \subseteq \mathsf{I}_{G'_\bot}$. As above $(\mathsf{I}_{G'_\bot} \setminus \mathsf{I}_{G'_{\mathsf{nil}}}) \cup H \in \mathbf{C}_{SP}$. By induction $G' \in \mathbf{C}_{SP}$. Since $\mathbf{0} \notin \mathrm{dom}(\mathsf{N}_{\mathsf{I}_{H_\top}})$ we get $S(G', (\mathsf{I}_{G_\bot} \setminus \mathsf{I}_{G'_{\mathsf{nil}}}) \cup H)$, so $G' \cup H \in \mathbf{C}_{SP}$.

Finally, since $S(G,H) \vee P(G,H) \Longrightarrow L(G,H)$, $\mathbf{C}_{SP} \subseteq \mathbf{C}_L$ by induction. □

2.5 *Expressible properties*

To end this section, we discuss some properties expressible with pi-charts. We may see the edges of a chart G as a relation $\rightarrow_G \subseteq \mathbf{I} \times \mathbf{L} \times \mathbf{I}$. We split this relation, writing $\iota_1 \xrightarrow{\mathtt{on}}_G \iota_2$ for $\exists \widetilde{c}, a.\ \iota_1 \xrightarrow{\langle \widetilde{c} \rangle\ \mathtt{on}\ a}_G \iota_2$ and $\iota_1 \downarrow\!\!\!\downarrow \iota_2$ for $\mathsf{E}_G(\iota_1,\iota_2) \in \mathbf{nL}$. Hence, we define some causal relations, roughly following the terminology of Priami [38].

Causal Relations:

$$[\text{Ancestor}]_G := \downarrow\!\!\!\downarrow^*_G \qquad [\text{Causes}]_G := (\downarrow\!\!\!\downarrow_G \cup \xrightarrow{\mathtt{on}}_G)^* \qquad [\text{Enables}]_G := (\downarrow\!\!\!\downarrow_G \cup \xrightarrow{\mathtt{on}}_G \cup (\xrightarrow{\mathtt{on}}_G)^{-1})^*$$

The node receiving a message enables the sending node and all of its descendants. This is due to the synchronous nature of communication: the sender proceeds with the knowledge that the message was received, just as if they had received an explicit acknowledgement of reception. The "causes" relation only flows in the direction of communicated messages; it is the equivalent in our setting to Lamport's "happened before" relation [26].

Another causal semantics for the pi-calculus is proved semantics [9,17,16], which makes a distinction between *subject* and *object* dependencies [7]. Since the latter are only defined in terms of "bound output" labels of a labelled transition system, they have no direct counterpart in our setting where all communication is internal to a pi-chart.

Let the nodes with ι as an ancestor be the *descendants* of ι. If ι_2, ι_3 are the nodes in the primitive chart for parallel composition, the sets of descendants of ι_2 and ι_3 are disjoint. (The "causes" and "enables" relations do not possess this property.)

Lemma 2.5 *If a pi-chart G has distinct edges $\iota_1 \xrightarrow{\varepsilon} \iota_2$ and $\iota_1 \xrightarrow{\varepsilon} \iota_3$ then there is no ι_4 such that both ι_2 $[Ancestor]_G$ ι_4 and ι_3 $[Ancestor]_G$ ι_4.*

We can concisely express some intensional properties of the interactions recorded in a chart G as follows (omitting the subscripts $_G$).

- "I got an answer to this message $(\iota_1 \xrightarrow{\langle a \rangle\ \mathtt{on}\ \widetilde{c}} \iota_2)$."

$$\exists \iota'.\ (\iota_1\ [\text{Ancestor}]\ \iota') \wedge (\iota_2\ [\text{Ancestor}] \xrightarrow{\mathtt{on}} \iota')$$

- "Every $end(t)$ event was caused by a corresponding $begin(t)$ event." [20]

$$\forall t, \iota_1, \iota_2 \exists \iota'.\ (\iota_1 \xrightarrow{end(t)} \iota_2) \Longrightarrow (\iota' \xrightarrow{begin(t)} [\text{Causes}]\ \iota_1)$$

- "I (ι_1) only communicated with descendants of somebody else (ι_2)."

$$\forall \iota' (\iota_1\ [\text{Ancestor}] \xrightarrow{\mathtt{on}} \iota') \Longrightarrow (\iota_2\ [\text{Ancestor}]\ \iota')$$

- "No name created by me (ι_1) was ever transmitted to somebody else (ι_2)."

$$\neg \exists \iota'_1, \iota'_2, a, \widetilde{c}.\ (b \in \widetilde{c}) \wedge (\iota_1\ [\text{Ancestor}] \xrightarrow{\mathtt{new}\ b} \iota'_1) \wedge (\iota'_1\ [\text{Ancestor}] \xrightarrow{\langle \widetilde{c} \rangle\ \mathtt{on}\ a} \iota'_2) \wedge (\iota_2\ [\text{Ancestor}]\ \iota'_2)$$

3 Relating the Reduction Semantics and Chart Semantics

Appendix A defines a standard reduction semantics for our pi-calculus [30,39], based on structural equivalence $P \equiv Q$, and reduction $P \to Q$. The only noteworthy detail is that constant instantiation is a rule of reduction, not structural equivalence. This avoids the syntactic constraints on definitions usually needed to avoid any unbounded unfolding.

3.1 Operational correspondence

We now develop the correspondence between the reduction semantics of the pi-calculus and pi-charts. We begin by defining the process corresponding to a pi-chart: the parallel composition of the processes at the bottom of the chart inside a restriction of the names generated in the chart.

Unloading a pi-chart G to a process: $[\![G]\!]$

$$[\![G]\!] := (\nu \mathsf{new}(G))(\prod_{\iota \in \mathsf{I}_{G_\perp}} \mathsf{N}_G(\iota)) \qquad \text{(hence: } [\![G]\!] = (\nu \mathsf{new}(G))[\![G_\perp]\!])$$

We split the primitive charts into housekeeping charts, that do not correspond to reduction steps, and computation charts, that do. Let the set of *housekeeping charts*, $\mathbf{C}_h$, be the subset of $\mathbf{C}_p$ generated just from the schemas for parallel composition and restriction. Let the set of *computation charts*, $\mathbf{C}_c$, be $\mathbf{C}_p \setminus \mathbf{C}_h$. Similarly, we split the chart extension relation $\to$ into two relations $\to_h$ and $\to_c$ as follows. If $G \to G \cup H$ with $H \in \mathbf{C}_h$, we write $G \to_h G \cup H$. Similarly $G \to_c G \cup H$ if $G \to G \cup H$ with $H \in \mathbf{C}_c$.

We can then show that housekeeping extension does not change the process corresponding to the chart, up to structural equivalence.

Lemma 3.1 *Suppose G is a pi-chart and $P \equiv [\![G]\!]$. If $G \to_h G \cup H$ then $P \equiv [\![G \cup H]\!]$.*

Reductions, on the other hand, are matched one for one by computation extension of charts, possibly with some housekeeping beforehand to reveal the redex.

Lemma 3.2 *Suppose G is a pi-chart and $P \equiv [\![G]\!]$.*

(1) *If $P \to P'$ then $G \to_h^* \to_c G'$ with $[\![G']\!] \equiv P'$.*

(2) *If $G \to_c G'$ then $P \to P'$ with $[\![G']\!] \equiv P'$.*

The full correspondence between many-step reduction of processes and pi-charts is then given by the following theorem.

Theorem 3.3 *$P \to^* Q$ iff there is a pi-chart G with $P \equiv [\![G_\top]\!]$ and $Q \equiv (\nu \mathsf{new}(G))[\![G_\perp]\!]$.*

Proof. By Lemma 3.1, Lemma 3.2 and induction, with G and G' in the lemmas given by $G := G_\top$ and $G' := G$. □

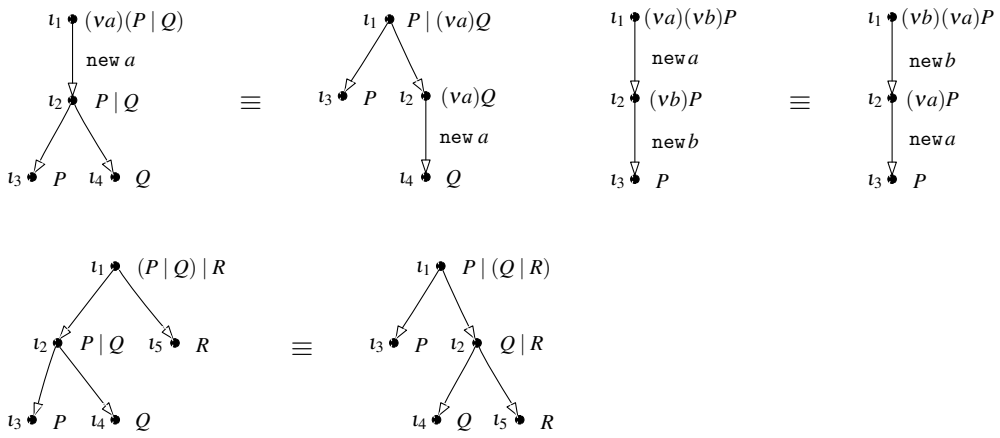
Conversely, if $[\![G_\top]\!] \to^* (\nu \mathsf{new}(G))[\![G_\perp]\!]$ for some graph G, the graph is not necessarily a pi-chart. It may have spurious edges, for example. We cannot expect to recover the notion of a pi-chart simply from the reduction semantics.

Many standard equivalences, such as barbed equivalence and congruence, are defined in terms of the relation $P \to^* Q$, plus direct observations of process structure [39]. Theorem 3.3 provides a basis for re-defining such equivalences in terms of charts.

3.2 *Structural equivalence on graphs*

The set of pi-charts $\{G \mid \iota \bullet P \rightarrow^* G\}$ generated by a process P is not preserved by structural congruence of processes, that is, it is not true that if $P \equiv Q$ then P and Q will generate the same charts, or even of the same shape. For example, consider two equivalent processes $(\nu a)(P \mid Q)$ and $P \mid (\nu a)Q$, where $a \notin \mathrm{fn}(P)$. The first process will generate a fresh name a and then branch to P and Q, whereas the second process will branch to P and $(\nu a)Q$, which then can generate the fresh name a. We reconcile these differences by defining a notion of structural congruence on graphs. Let $G \equiv G'$ be the least relation on graphs that is reflexive, symmetric and transitive and that satisfies the following axioms.

Structural Equivalence on Graphs: $G \equiv H$

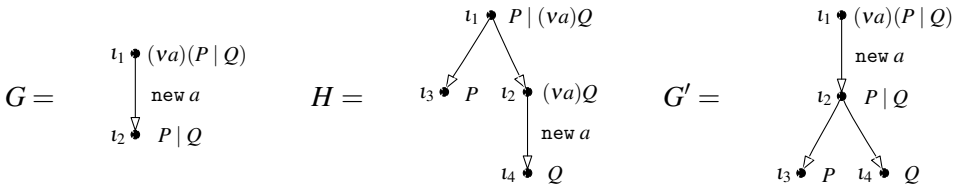


$$\frac{P \equiv Q}{\iota \bullet P \equiv \iota \bullet Q} \qquad \frac{G \equiv G' \quad H \equiv H' \quad L(G,H) \quad L(G',H')}{G \cup H \equiv G' \cup H'}$$

We note various properties of equivalent charts in Lemma 3.4. The lemma states that the nodes at the top of equivalent charts are equal up to structural congruence of processes (1), and similarly for the nodes at the bottom (2). Equivalent charts also generate the same fresh names (3), and their corresponding processes are equivalent (4).

Lemma 3.4 *If G is a pi-chart and $G \equiv H$ then H is a pi-chart and (1) $G_\top \equiv H_\top$, (2) $G_\bot \equiv H_\bot$, (3) $\mathsf{new}(G) = \mathsf{new}(H)$, and (4) $[\![G]\!] \equiv [\![H]\!]$.*

We can characterize structural equivalence of processes in terms of the extension relation $G \rightarrow G'$ on graphs and structural equivalence of graphs, as stated in Theorem 3.5. The theorem states that equivalent processes generate equivalent charts, up to housekeeping extensions. Ideally we would like the statement of Theorem 3.5 to hold for $G \equiv H$ rather than the weaker $G \rightarrow_h^* \equiv H$ (recall that $\rightarrow_h$ denotes a "housekeeping" transition involving a parallel composition or a restriction). Unfortunately, the stronger statement does not hold in general. For a counterexample, consider the two equivalent processes $(\nu a)(P \mid Q)$ and $P \mid (\nu a)Q$, where $a \notin \mathrm{fn}(P)$, and the charts G, H, G' defined as follows.



We have that $\iota_1 \bullet (\nu a)(P \mid Q)$ can extend to G but $\iota_1 \bullet P \mid (\nu a)Q$ cannot extend to any chart structurally congruent to G. However, $\iota_1 \bullet P \mid (\nu a)Q \longrightarrow\!\!\!\!\rightarrow H$ and G can perform an additional housekeeping extension to G' with $G' \equiv H$.

Theorem 3.5 *$P \equiv Q$ iff whenever $\iota \bullet P \rightarrow^* G$ there is H with $\iota \bullet Q \rightarrow^* H$ and $G \rightarrow_h^* \equiv H$.*

4 Related Work

Starting with Petri [34], there is a substantial literature on graphs as a notation for states of concurrent computations. Examples include process algebras inspired by Petri Nets [3], together with a range of graph-based notations such as [18] and its numerous citations. In the area of process calculi Milner's pi-nets [29] represent pi-calculus processes as graphs, where each node represents a channel and edges to a node represent inputs or outputs on the channel. Rewrite rules on graphs coalesce nodes after an interaction. Other graph-rewriting based models for the pi calculus include a hypergraph semantics [25] and a term graph semantics [18]. History dependent automata [32] map the entire state space of a pi-calculus process, where each node represents a separate state. The history of names is recorded in the graph, but not the history of computations. Bigraphs [31] are a graphical representation of both the computational and spatial aspects of a process. The graphical stochastic pi-calculus [35] represents a pi-calculus process as a collection of synchronising automata. All these process representations use graphs to represent states of computations, but not the computation history. (However, in certain of the cases one can recover causal relationships [10].) In contrast, a pi-chart represents one of the possible interaction histories of a set of processes, themselves given by syntax trees.

A trace is a sequence of actions performed by a process. In the setting of the pi-calculus, there are several formal definitions of trace, with the aim of defining properties of type systems [39], investigating asynchronous equivalences [6], and defining correspondence assertions [20]. Proved traces [8,38] are decorated with the locations in the term that participated in a transition. Pi-charts enable two-dimensional rendering and record more information, especially regarding restricted names as the subjects and objects of communication.

Various graphical structures are used to define noninterleaving semantics and equivalences of processes; this work has mainly concerned other process calculi and algebras, but recently Varacca and Yoshida [41] develop such a semantics for the pi-calculus using event structures [42]. In contrast, pi-charts are not directly useful (and are not intended) for generating equivalences on processes. The equivalence induced by the set of pi-charts $\{G \mid \iota \bullet P \rightarrow^* G\}$ extending from a process P is syntactic identity, since the process P is embedded in each member of the set. Of course, Theorem 3.3 allows us to reformulate any equivalence relation defined using the interleaving semantics $P \rightarrow^* Q$ in terms of the chart semantics. Our development of structural congruence of graphs, leading to Theorem 3.5, begins the study of equivalences induced by charts.

Cryptographic security protocols are often specified by protocol narrations [1], exemplary sequences of communications of the form "Message n $X \rightarrow Y$: M", meaning that the nth message M of the protocol goes from role X to role Y. A narration itself is essentially an MSC. Some formalisms for security protocols represent protocol runs as MSCs, essentially.

For example, strand spaces [40] are a graphical formalism for protocol narrations, based on strands and bundles. Each strand is a string of inputs and outputs, with implicit name generation, representing a role in the protocol. A bundle is a directed acyclic graph obtained by composing strands, similar to an MSC. Properties of protocols are expressed in terms of occurrences of strands within bundles and "ancestor of" and "earlier than" relations, similar to the causal relations in Section 2.

Crazzolara and Milicia [14] establish explicit formal links between MSCs, formalized as pomsets [37], and the semantics of the Security Protocol Language (SPL) [15]. SPL can be seen as a simple process calculus, with broadcast communication, but without process forking as in the pi-calculus. They define an algorithm for constructing an MSC from any finite trace in the transition semantics of an SPL program. Their main formal result is that the events of such an MSC can be linearized to match the trace and moreover that every linearization of the MSC corresponds to a trace of the original SPL program. Their MSCs are extracted from an existing semantics for SPL, rather than being defined directly.

5 Conclusion

To summarize, our chart semantics is the first semantics for the pi-calculus based on the idea of message sequence charts. The main benefits of pi-charts compared to a conventional relational semantics are: (1) pi-charts are easier to visualize; and (2) pi-charts can express ancestry and causal dependencies that state-based relational semantics omit.

Although a chart corresponds to a single execution trace, in future we envisage verification tools for proving properties about the set of all charts generated by a given process. For example, this could be useful for validating high-level protocols expressed as pi-calculus processes. In cases where the desired properties do not hold, a visual execution trace representing a counter-example could be presented to the user.

The pi-calculus is used to model programming language features, communication and security protocols and their properties, and more recently, aspects of systems biology (see Appendix B for an example). Hence, the broader significance of our work beyond the pi-calculus is that it forms a formal basis to help visualize and express properties of systems in all of these areas.

Acknowledgements

U. Nestmann and G. Winskel advised us on related work. J. Guttman helped us understand the connection between pi-charts and strand spaces.

A Standard Relational Semantics

Structural Equivalence: $P \equiv Q$ and Reduction: $P \to Q$

$$P \equiv P \qquad P \to P' \Rightarrow P \mid Q \to P' \mid Q$$

$$Q \equiv P \Rightarrow P \equiv Q \qquad P \to P' \Rightarrow (\nu a)P \to (\nu a)P'$$

$$P \equiv Q, Q \equiv R \Rightarrow P \equiv R \qquad P \equiv Q, Q \to Q', Q' \equiv P' \Rightarrow P \to P'$$

$$P \equiv P' \Rightarrow (\nu x)P \equiv (\nu x)P' \qquad (a(\widetilde{x}).P + M) \mid (\overline{a}\langle\widetilde{c}\rangle.Q + N) \to P\left\{{}^{\widetilde{c}}/_{\widetilde{x}}\right\} \mid Q$$

$$P \equiv P' \Rightarrow P \mid Q \equiv P' \mid Q \qquad \tau_t.P + M \to P$$

$$P \mid Q \equiv Q \mid P \qquad \mathsf{A}(\widetilde{c}) \to P\left\{{}^{\widetilde{c}}/_{\widetilde{x}}\right\} \text{ if } \mathsf{A}(\widetilde{x}) := P$$

$$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$$

$$a \notin \mathrm{fn}(P) \Rightarrow (\nu a)(P \mid Q) \equiv P \mid (\nu a)Q$$

$$(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$$

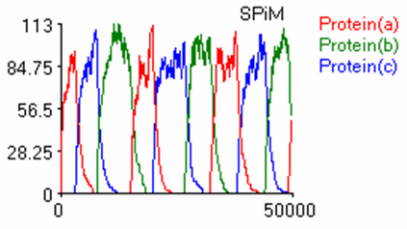
As stated in Section 2, we identify processes up to associativity and commutativity of the choice operator.

B A Biological Example

This example shows how pi-charts can be a useful tool for visualising interactions between stochastic pi-calculus models of biological systems. We use the same pi-calculus syntax and reduction rules from Sections 2 and 3, enriched with a stochastic extension along the lines of [35]. Stochastic behaviour is incorporated into the calculus by associating each channel a with a corresponding interaction rate given by $\rho(a)$, and associating each action τ_r with a delay rate r. The rates are used as the basis for a stochastic simulation algorithm, which calculates the probability of all possible reductions at each step and stochastically chooses the next reduction based on these probabilities.

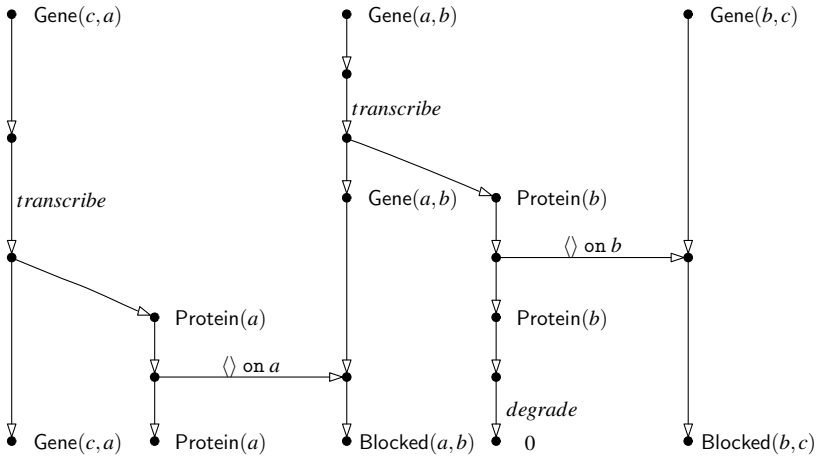
Consider the following network of three genes that mutually repress each other, with definitions for $\mathsf{Gene}(a,b)$, $\mathsf{Blocked}(a,b)$ and $\mathsf{Protein}(b)$ based on [35]:

$$\begin{aligned}
&\mathsf{Gene}(a,b) := \tau_{transcribe}.(\mathsf{Gene}(a,b) \mid \mathsf{Protein}(b)) \\
&\qquad\qquad + a().\mathsf{Blocked}(a,b) \\
&\mathsf{Blocked}(a,b) := \tau_{unblock}.\mathsf{Gene}(a,b) \\
&\mathsf{Protein}(b) := \overline{b}\langle\rangle.\mathsf{Protein}(b) \\
&\qquad\qquad + \tau_{degrade} \\
&\\
&\mathsf{Gene}(a,b) \mid \mathsf{Gene}(b,c) \mid \mathsf{Gene}(c,a)
\end{aligned}$$



The $\mathsf{Gene}(a,b)$ is parameterised by its promoter region a, together with the promoter region b that is recognised by its transcribed proteins. The gene can perform one of two actions. Either it can transcribe a $\mathsf{Protein}(b)$ by doing a stochastic delay at rate *transcribe*, after which the new protein is executed in parallel with the gene, or it can block by doing an input on its promoter region a. The blocked gene can then unblock by doing a stochastic delay at rate *unblock*. The $\mathsf{Protein}(b)$ can repeatedly do an output on the pro-

moter region b, or it can decay at rate *degrade*. According to the reduction rules of the calculus, the output $\overline{b}\langle\rangle$ of the transcribed protein can interact with the input $b()$ of a $\mathsf{Gene}(b,c)$, which becomes blocked as a result. The three genes $\mathsf{Gene}(a,b)$, $\mathsf{Gene}(b,c)$ and $\mathsf{Gene}(c,a)$ can mutually repress each other, since $\mathsf{Gene}(a,b)$ produces a protein that can block $\mathsf{Gene}(b,c)$, which produces a protein that can block $\mathsf{Gene}(c,a)$, which produces a protein that can block $\mathsf{Gene}(a,b)$, completing the cycle. This mutual repression gives rise to alternate oscillation of protein levels, as shown in the above simulation plot, in which the vertical axis represents the number of processes and the horizontal axis represents the simulation time. The results were obtained with equal rates for channels a,b,c such that $\rho(a) \gg \mathit{transcribe} \gg \mathit{degrade} > \mathit{unblock}$. However, the plots themselves give no indication as to what actually causes the oscillations to occur. Such a question is fundamental to understanding the behaviour the system, and pi-charts can help to provide a partial answer. An execution trace for the system is represented by the following pi-chart, which shows how the system can evolve starting from one of each gene. The visual representation of causality in the pi-chart helps to clarify the sequence of execution steps leading to the first oscillation cycle.



The chart shows how one of the genes, in this case $\mathsf{Gene}(a,b)$, transcribes a $\mathsf{Protein}(b)$, which immediately blocks $\mathsf{Gene}(b,c)$. $\mathsf{Gene}(c,a)$ transcribes $\mathsf{Protein}(a)$ soon after, which blocks $\mathsf{Gene}(a,b)$.The $\mathsf{Gene}(a,b)$ and $\mathsf{Gene}(b,c)$ both remain blocked, waiting for a slow *unblock* delay to fire, while $\mathsf{Gene}(c,a)$ is able to freely produce $\mathsf{Protein}(a)$ and start the first oscillation cycle.

We have implemented a prototype stochastic simulator that automatically generates a pi-chart during a given simulation run. The prototype was implemented as a simple extension to the SPiM simulator, [5] by exporting the execution history of a simulation to a file using the DOT syntax [19]. The DOT layout engine is then used to automatically render the file as a pi-chart. The generated charts can be quite large, but it is relatively easily to scroll and zoom through the charts to a time point of particular interest in the simulation. For the above biological example one can focus on the sequence of transitions leading up to a switch in oscillation cycles, which can be quite informative.

In general, pi-charts seem to be a convenient way of visualising and debugging the

[5] SPiM is available at http://research.microsoft.com/~aphillip/spim/.

behaviour of concurrent biological systems, and initial reactions from biologists have so far been positive. We plan to include a pi-chart debugging option in the next release of the SPiM simulator, so that biologists can experiment with generating their own charts from a range of models.

C Expressing the Bounds Guaranteed by a Type System

We present a synthesis of some existing type systems, including groups (or sorts) [30], group creation [12], and usage bounds [24]. A channel type T takes the general form $g\ ?i\ !o\ [T_1,\ldots,T_n]$. We say g is the *group* of the type, and of names belonging to the type. Groups indicate different usages, for example, REQ or RES. A name x of type T is a channel conveying tuples of names with types T_1, ..., T_n. The *multiplicities* i and o are upper bounds on the number of uses of x for input and output.

Group creation $(\nu grp\ g)P$ makes a fresh group g for use within P. Groups are represented as names, but well-typed processes cannot send them on channels. Hence, group creation helps structure processes by confining the flow of names belonging to a group. In particular, if a process $O \mid (\nu grp\ g)P$ is well-typed and there is a name x in group g, then the name x is communicated only between descendants of P—the lexical scope of g—and cannot flow to descendants of O.

Our point here is not the type system itself, an assembly of variations of existing components, but rather to show that pi-charts can conveniently express both the usage bounds induced by multiplicities and the secrecy properties induced by group creation. The original statement of the latter [12, Proposition 3] relies on an informal notion of process derivation; our statement in terms of the "ancestor of" relation is completely formal.

We proceed with a terse presentation of the type system. Further explanations and examples are in the original publications [12,24,30].

Groups and Types:

g,h	group: subset of the set of the names
$\mu,i,o ::= 0 \mid 1 \mid \omega$	multiplicity
$T ::= g\ ?i\ !o\ [T_1,\ldots,T_n]$	polyadic channel type ($n \geq 0$)
$m ::= g \mid (x:T)$	item: either a group, or a name with a type
$E ::= \varnothing, m_1,\ldots,m_n$	typing environment: finite list of items

$\mathrm{dom}(\varnothing) := \varnothing \quad \mathrm{dom}(E,g) := \mathrm{dom}(E) \cup \{g\} \quad \mathrm{dom}(E,x:T) = \mathrm{dom}(E) \cup \{x\}$

Our pi-calculus syntax is untyped, but we place type and group annotations on τ prefixes, both to guide typechecking, and to record typing information in the pi-chart semantics. We take the algebra of annotations $\mathbf{A}$ to be the set of items, so that we can write $\tau_g.P$ and $\tau_{x:T}.P$. Let *typed name restriction* be $(\nu x:T)P := (\nu x)\tau_{x:T}.P$ and *group creation* be $(\nu grp\ g)P := (\nu g)\tau_g.P$. Every chart extending from $(\nu x:T)P$ and reaching P includes edges $\iota \xrightarrow{\texttt{new}\,x} \iota'$ and $\iota' \xrightarrow{x:T} \iota''$ and node $\iota'' \bullet P$. Similarly, every chart extending from $(\nu grp\ g)P$ and reaching P includes $\iota \xrightarrow{\texttt{new}\,g} \iota'$, $\iota' \xrightarrow{grp\ g} \iota''$, and $\iota'' \bullet P$.

Let the addition $\mu+\mu'$ of two multiplicities be the commutative function satisfying the equations $\mu+0=\mu$ and $\mu+\omega=\omega$ and $0+1=1$ and $1+1=\omega$. The addition functions on types, items, and environments are the least partial functions to satisfy the following equations. They are all associative and commutative.

Type, Item, and Environment Addition: $T+T' \quad m+m' \quad E+E'$

$$(g\ ?i\ !o\ [T_1,\ldots,T_n]) + (g\ ?i'\ !o'\ [T_1,\ldots,T_n]) := g\ ?(i+i')\ !(o+o')\ [T_1,\ldots,T_n]$$
$$g+g := g$$
$$(x:T)+(x:T') := x:(T+T')$$
$$(\varnothing, m_1,\ldots,m_n)+(\varnothing, m'_1,\ldots,m'_n) := (\varnothing, m_1+m'_1,\ldots,m_n+m'_n)$$

We assume a relation between process constants and lists of groups and types describing their parameters. Specifically, for each definition $\mathsf{A}(x_1,\ldots,x_n) := P$, we assume that the constant A is related to a list of group parameters $h_1,\ldots,h_m$ and a list of types $T_1,\ldots,T_n$. We write this as $\mathsf{A}[h_1,\ldots,h_m,x_1:T_1,\ldots,x_n:T_n]$.

The following rules define four judgments: $E \vdash \diamond$ means that the environment E is well-formed; $E \vdash m$ means that the item m occurs in E; $E \vdash T$ means that the type T is well-formed in E; and $E \vdash P$ means that the process P is well-formed in E.

Typing Rules: $E \vdash \diamond \quad E \vdash m \quad E \vdash T \quad E \vdash P$

$$\frac{}{\varnothing \vdash \diamond} \qquad \frac{E \vdash \diamond \quad g \notin \mathrm{dom}(E)}{E,g \vdash \diamond} \qquad \frac{E \vdash T \quad x \notin \mathrm{dom}(E)}{E,x:T \vdash \diamond}$$

$$\frac{\varnothing, m_1,\ldots,m_n \vdash \diamond \quad i \in 1..n}{\varnothing, m_1,\ldots,m_n \vdash m_i} \qquad \frac{E \vdash g \quad E \vdash T_1 \quad \ldots \quad E \vdash T_n}{E \vdash g\ ?i\ !o\ [T_1,\ldots,T_n]} \qquad \frac{E \vdash \diamond}{E \vdash \mathbf{0}}$$

$$\frac{E_0 \vdash x : g\ ?1\ !0\ [T_1,\ldots,T_n] \quad E_1, y_1:T_1,\ldots,y_n:T_n \vdash P \quad E = E_0+E_1 \text{ defined}}{E \vdash x(y_1,\ldots,y_n).P}$$

$$\frac{E_0 \vdash x : g\ ?0\ !1\ [T_1,\ldots,T_n] \quad E_i \vdash y_i : T_i \quad \forall i \in 1..n \quad E_{n+1} \vdash P \quad E = E_0+\cdots+E_{n+1} \text{ defined}}{E \vdash \overline{x}\langle y_1,\ldots,y_n\rangle.P} \qquad \frac{E \vdash m \quad E \vdash P}{E \vdash \tau_m.P} \qquad \frac{E \vdash M \quad E \vdash N}{E \vdash M+N}$$

$$\frac{E,g \vdash P}{E \vdash (\nu grp\ g)P} \qquad \frac{E,x:T \vdash P}{E \vdash (\nu x:T)P} \qquad \frac{E_1 \vdash P_1 \quad E_2 \vdash P_2 \quad E = E_1+E_2 \text{ defined}}{E \vdash P_1 \mid P_2}$$

$$\frac{\mathsf{A}[h_1,\ldots,h_m,x_1:T_1,\ldots,x_n:T_n] \quad \sigma = \{{}^{g_j}/_{h_j} \mid j \in 1..m\} \quad E \vdash \diamond \quad E \vdash g_j \quad \forall j \in 1..m \quad E_i \vdash c_i : T_i\sigma \quad \forall i \in 1..n \quad E = E_1+\cdots+E_n \text{ defined}}{E \vdash \mathsf{A}(c_1,\ldots,c_n)}$$

We assume that $h_1,\ldots,h_m,x_1:T_1,\ldots,x_n:T_n \vdash P$ for each definition $\mathsf{A}(x_1,\ldots,x_n) := P$ where $\mathsf{A}[h_1,\ldots,h_m,x_1:T_1,\ldots,x_n:T_n]$.

Theorem C.1 *Suppose $E \vdash [\![G_\top]\!]$, G is a pi-chart, and $T = g\ ?i\ !o\ [T_1,\ldots,T_n]$.*

(1) *If $\iota_1 \xrightarrow{x:T} \iota_2$ then the number of communications on x in G is no more than $\min(i,o)$.*

(2) *If $\iota_1 \xrightarrow{g} \iota_2$ and $\iota_3 \xrightarrow{x:T} \iota_4$ then ι_2 [Ancestor] ι_3.*
Moreover, if $\iota_5 \xrightarrow{\langle\widetilde{y}\rangle\ \mathtt{on}\ z} \iota_6$ and $x \in \mathrm{fn}(\widetilde{y},z)$ then ι_2 [Ancestor] ι_5 and ι_2 [Ancestor] ι_6.

We can explain the secrecy property of group creation by appeal to this theorem. Suppose that $E \vdash O \mid (\nu grp\ g)P$, and consider any pi-chart G such that $G_\top = \iota \bullet (O \mid (\nu grp\ g)P)$ for some ι. Such a G represents an arbitrary interaction between the process O and the process $(\nu grp\ g)P$. Unless G is a singleton, in which case it includes no interactions, it must include an instance of the primitive chart for parallel composition, with edges $\iota \rightarrowtriangle \iota'$ and $\iota \rightarrowtriangle \iota''$, and nodes $\iota' \bullet O$ and $\iota'' \bullet (\nu grp\ g)P$. As discussed above, if P is reached, there must be edges $\iota'' \xrightarrow{\texttt{new}\ g} \iota_1$, $\iota_1 \xrightarrow{grp\ g} \iota_2$, and a node $\iota_2 \bullet P$. By Lemma 2.5, no descendant of $\iota' \bullet O$ is a descendant of $\iota_2 \bullet P$, and the converse. If a name x of group g is created, there must be an edge $\iota_3 \xrightarrow{x:T} \iota_4$, where g is the group of T. By Theorem C.1(ii), ι_2 [Ancestor] ι_3, that is, a descendant of P creates the name x. Now, consider any communication of x, that is, consider any edge $\iota_5 \xrightarrow{\langle \widetilde{y} \rangle\ \texttt{on}\ z} \iota_6$ with $x \in \mathrm{fn}(\widetilde{y})$. By Theorem C.1(ii), ι_2 [Ancestor] ι_5 and ι_2 [Ancestor] ι_6, that is, both the sender ι_5 and the receiver ι_6 of the tuple $\widetilde{y}$ containing x are descendants of P. Additionally, the theorem implies that all communications on the channel x itself are between descendants of P.

Hence, pi-charts directly formalize the intention that "channels of group g are forever secret outside the initial scope of $(\nu grp\ g)$" [12].

References

[1] M. Abadi. Security protocols and their properties. In *Foundations of Secure Computation*, NATO Science Series, pages 39–60. IOS Press, 2000.

[2] R. Alur, G. J. Holzmann, and D. Peled. An analyzer for message sequence charts. *Software Concepts and Tools*, 17(2):70–77, 1996.

[3] E. Best, R. R. Devillers, and M. Koutny. The box algebra = petri nets + process expressions. *Inf. Comput.*, 178(1):44–100, 2002.

[4] K. Bhargavan, C. Fournet, A. D. Gordon, and R. Pucella. TulaFale: A security tool for web services. In *FMCO'03*, volume 3188 of *LNCS*, pages 197–222. Springer, 2004.

[5] E. Bonelli, A. Compagnoni, and E. Gunter. Correspondence assertions for process synchronization in concurrent communications. *JFP*, 15(2):219–147, 2004.

[6] M. Boreale, R. De Nicola, and R. Pugliese. Trace and testing equivalence on asynchronous processes. *Information and Computation*, 172(2):139–164, 2002.

[7] M. Boreale and D. Sangiorgi. A fully abstract semantics of causality in the π-calculus. *Acta Informatica*, 35:353–400, 1998.

[8] G. Boudol and I. Castellani. Concurrency and atomicity. *TCS*, 59:25–84, 1988.

[9] G. Boudol and I Castellani. A non-interleaving semantics for CCS based on proved transitions. *Fundamenta Informaticae*, 4(XI):433–452, 1988.

[10] R. Bruni, H. C. Melgratti, and U. Montanari. Event structure semantics for nominal calculi. In *CONCUR*, pages 295–309, 2006.

[11] M. Bugliesi, G. Castagna, and S. Crafa. Boxed ambients. In *Theoretical Aspects of Computer Software (TACS 2001)*, volume 2215 of *LNCS*, pages 38–63. Springer, 2001.

[12] L. Cardelli, G. Ghelli, and A. D. Gordon. Secrecy and group creation. *Information and Computation*, 196(2):127–155, 2005.

[13] I. Castellani. Process algebras with localities. In *Handbook of Process Algebra*, chapter 15, pages 945–1045. Elsevier, 2001.

[14] F. Crazzolara and G. Milicia. Graphical descriptions of security protocols. In *COnstraint & LOgic Programming in Security (COLOPS 2003)*, 2003.

[15] F. Crazzolara and G. Winskel. Events in security protocols. In *Eight ACM Conference on Computer and Communications Security (CCS'2001)*, 2001.

[16] P. Degano and C. Priami. Non interleaving semantics for mobile processes. *Theoretical Computer Science*, 216:237–270, 1999.

[17] P. Degano and C. Priami. Enhanced operational semantics. *ACM Computing Surveys*, 2(33):135–176, 2001.

[18] F. Gadducci. Term graph rewriting for the pi-calculus. In Atsushi Ohori, editor, *APLAS*, volume 2895 of *Lecture Notes in Computer Science*, pages 37–54. Springer, 2003.

[19] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software—Practice and Experience*, 30(11):1203–1233, 2000.

[20] A. D. Gordon and A. Jeffrey. Typing correspondence assertions for communication protocols. *Theoretical Computer Science*, 300:379–409, 2003.

[21] ITU. *Message Sequence Chart (MSC)*, 1999. Recommendation Z.120.

[22] ITU. *Specification and Design Language (SDL)*, 1999. Recommendation Z.100.

[23] A. S. A. Jeffrey and J. Rathke. A fully abstract may testing semantics for concurrent objects. *Theoretical Computer Science*, 338:17–63, 2005.

[24] N. Kobayashi, B. Pierce, and D. Turner. Linearity and the pi-calculus. *TOPLAS*, 21(5):914–947, 1999.

[25] B. König. A graph rewriting semantics for the polyadic pi-calculus. In *Proc. of GT-VMT '00 (Workshop on Graph Transformation and Visual Modeling Techniques)*, pages 451–458. Carleton Scientific, 2000.

[26] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[27] S. Mauw and M. A. Reniers. An algebraic semantics of basic message sequence charts. *The Computer Journal*, 37(4), 1994.

[28] S. Mauw, M.A. Reniers, and T.A.C. Willemse. Message Sequence Charts in the software engineering process. In S.K. Chang, editor, *Handbook of Software Engineering and Knowledge Engineering*. World Scientific, 2000.

[29] R. Milner. Pi-nets: A graphical form of π-calculus. In *ESOP'94*, pages 26–42, 1994.

[30] R. Milner. *Communicating and Mobile Systems: the π-Calculus*. CUP, 1999.

[31] R. Milner. Bigraphical reactive systems: Basic theory. Technical Report 523, University of Cambridge Computer Laboratory, 2001.

[32] U. Montanari and M. Pistore. History-dependent automata: An introduction. *Lecture Notes in Computer Science*, 3465:1–28, 2005.

[33] Object Management Group. *Unified Modeling Language*. At `http://www.uml.org`.

[34] C. Petri. Fundamentals of a theory of asynchronous information flow. In *IFIP Congress '62*, pages 386–390. North Holland, 1962.

[35] A. Phillips, L. Cardelli, and G. Castagna. A graphical representation for biological processes in the stochastic pi-calculus. *Transactions in Computational Systems Biology*, 4230:123–152, 2006.

[36] A. Phillips, N. Yoshida, and S. Eisenbach. A distributed abstract machine for boxed ambient calculi. In *European Symposium on Programming*, LNCS. Springer, April 2004.

[37] V. Pratt. Modelling concurrency with partial orders. *International Journal of Parallel Programming*, 15(1):33–71, 1986.

[38] C. Priami. *Enhanced Operational Semantics for Concurrency*. PhD thesis, Pisa, 1996.

[39] D. Sangiorgi and D. Walker. *The π-calculus: A theory of mobile processes*. CUP, 2001.

[40] F. J. Thayer Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7:191–230, 1999.

[41] D. Varacca and N. Yoshida. Typed event structures and the pi-calculus. In *Mathematical Foundations of Programming Semantics*, ENTCS. Elsevier, 2006.

[42] G. Winskel. *Events in Computation*. PhD thesis, University of Edinburgh, 1980.

Matching Systems for Concurrent Calculi

Bjørn Haagensen[1]

Aalborg University
Denmark

Sergio Maffeis[2]

Imperial College London

Iain Phillips[3]

Imperial College London

Abstract

Matching systems were introduced by Carbone and Maffeis, and used to investigate the expressiveness of the pi-calculus with polyadic synchronisation. We adapt their definition and investigate matching systems for CCS, the pi-calculus and Mobile Ambients. We show among other results that the asynchronous pi-calculus with matching cannot be encoded (under certain conditions) in CCS with polyadic synchronisation of all finite levels.

Keywords: Matching systems, CCS, pi-calculus, Mobile Ambients

1 Introduction

Matching systems were introduced by Carbone and Maffeis [4]. A matching system is a protocol which ensures that a client matches successfully with a server if and only if both parties have the same sequence of names as parameters. This can be achieved trivially if client and server can synchronise on all names in a single atomic communication. However it may not be possible if, for instance, they can only synchronise on one name at a time, as in standard π-calculus. Carbone and Maffeis used matching systems to establish a hierarchy within ${}^{e}\pi$, the π-calculus with polyadic synchronisation. They show that there is no encoding (satisfying certain conditions) of the asynchronous calculus with n-adic communication into the synchronous calculus with m-adic communication (for any $m < n$).

[1] Email: bh@cs.aau.dk
[2] Email: maffeis@doc.ic.ac.uk
[3] Email: iccp@doc.ic.ac.uk

This is a preliminary version. The final version will be electronically published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.com/locate/entcs

In this paper we investigate matching systems further. In particular, we propose a weakened form of matching system, where if client and server agree on their parameters then there is a successful computation, but success is no longer guaranteed, unlike in the original formulation. These weak matching systems enable us to obtain different separations between calculi. We regard matching systems (whether in the weak form or the original strong form) as measuring the capability of particular calculi to perform transactions, in other words to perform a series of operations which can be treated as a single operation. Weak matching systems require only that all the commits are justified, whereas strong matching systems require also that the transaction is not rolled back an unbounded number of times.

In [4], it is shown that there is no "sensible" encoding of matching in the π-calculus with mixed choice. Here we use weak matching systems to give a different separation result involving a language with matching, based on a different notion of encoding. We shall show there there is no encoding (subject to certain conditions) from the asynchronous π-calculus with matching into CCS with n-adic communication (for any n).

This is related to the question of showing that the asynchronous π-calculus cannot be encoded into CCS. As far as we are aware, such a negative result has never been obtained, even though most researchers would presumably expect this to hold, since the asynchronous π-calculus has the ability to send and receive names (objects) and then use them as channels (subjects), and this is disallowed in CCS (even with value passing).

Palamidessi [9] used electoral systems to prove two relevant results. Firstly, she showed that CCS cannot be encoded in the asynchronous π-calculus (the converse of what we are discussing). Secondly, she showed that the π-calculus with mixed choice cannot be encoded in CCS. However her work leaves open the possibility that the asynchronous π-calculus can be encoded in CCS.

Banach and van Breugel [1] encoded the π-calculus into a version of CCS. This involves augmenting CCS with infinite operations (and not just infinite summation).

Sangiorgi [12] defined the π-calculus with internal mobility (πI), where only private names can be transmitted. He gave a hierarchy of typed calculi within πI, such that the bottom level represents "the core of CCS". He showed that higher levels in the hierarchy exhibit a "higher degree" of mobility, in the sense that they admit longer subject-object dependency chains. However he did not assert any result about the non-encodability of higher levels in lower levels of the hierarchy.

Boreale [2] gave an encoding of the asynchronous π-calculus into πI. This encoding is in two steps, and goes via an intermediate calculus, *localised* π, or Lπ, the subset of the asynchronous π-calculus where the recipient of a name may only use it in output actions. This terminology is due to Merro and Sangiorgi [7]. They showed that Lπ can be encoded fully abstractly in localised πI using the second step in Boreale's encoding.

After presenting our results on weak matching systems, we recast the separation result concerning polyadic synchronisation of [4] into our current setting, using the notion of replicated strong matching systems. Our new formulation is a slight generalisation of the previous result. Surprisingly, and against previous intuition, we have found that by simply requiring each instance of a matching system to be

finite (strong matching systems), the full π-calculus is powerful enough to solve the problem for any degree n. We conjecture that the same is not possible for the asynchronous π-calculus, suggesting a possible new interpretation of the expressive power of the mixed choice construct.

The remainder of the paper is organised as follows. In Section 2 we define the calculi we shall be considering. Then in Sections 3 and 4 we investigate weak and strong matching systems, respectively. We finish with conclusions and further work.

2 Calculi

In this section we define the calculi that we shall be concerned with in this paper.

We let $x, y, \ldots$ range over the set of *names* $\mathcal{N}$. We shall let $\overrightarrow{x}$ denote a tuple of names $x_1, \ldots, x_n$.

Polyadic synchronisation, where e.g. an output process $\overline{x \cdot y}\langle z\rangle.P$ can synchronise with an input process $x \cdot y(w).Q$, was introduced in [4].

Definition 2.1 *The full π_n-calculus ($\mathrm{f}\pi_n$) is defined as the polyadic synchronous π-calculus with mixed choice, matching and mismatching, and polyadic synchronisation of degree n, that is*

$$P ::= P \mid Q \;\mid\; \nu x P \;\mid !P \;\mid\; \Sigma_i \alpha_i.P \;\mid\; [x = y]P \;\mid\; [x \neq y]P$$

where each α_i is of the form $x_1 \cdot \ldots \cdot x_n(\overrightarrow{y})$ or $\overline{x_1 \cdot \ldots \cdot x_n}\langle \overrightarrow{y}\rangle$. We let S, T range over summations, and write the empty summation as $\mathbf{0}$.

Note that $\mathrm{f}\pi_1$ is the standard full π-calculus. We define the free names $\mathsf{fn}(P)$ as usual, with input and restriction being name-binding.

Definition 2.2 Structural congruence *is the least congruence $\equiv$ on $\mathrm{f}\pi_n$ processes satisfying the following laws: $P \mid Q \equiv Q \mid P$, $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$, $\mathbf{0} \mid P \equiv P$, $[x = x]P \equiv P$, $[x \neq y]P \equiv P$ if $x \neq y$, $!P \equiv P \mid !P$ and $\nu x(P \mid Q) \equiv P \mid \nu x Q$ if $x \notin \mathsf{fn}(P)$, together with reordering of summations.*

Definition 2.3 *The reduction relation on $\mathrm{f}\pi_n$ is defined by the following axiom and rules*

$$(\overline{x_1 \cdot \ldots \cdot x_n}\langle y_1, \ldots, y_m\rangle.P + S) \mid (x_1 \cdot \ldots \cdot x_n(z_1, \ldots, z_m).Q + T) \\ \rightarrow P \mid Q\{y_1, \ldots, y_m / z_1, \ldots, z_m\}$$

$$\frac{P \rightarrow Q}{P \mid R \rightarrow Q \mid R} \qquad \frac{P \rightarrow Q}{\nu x P \rightarrow \nu x Q} \qquad \frac{P' \equiv P \quad P \rightarrow Q \quad Q \equiv Q'}{P' \rightarrow Q'}$$

We let $\Rightarrow$ be the reflexive and transitive closure of $\rightarrow$.

Definition 2.4 *Input and output* barbs *are defined by*

$$P \downarrow_{x_1 \cdot \ldots \cdot x_n} \;\; \textit{iff } P \equiv \nu \overrightarrow{z}((x_1 \cdot \ldots \cdot x_n(\overrightarrow{y}).R + S) \mid Q) \textit{ where } \overrightarrow{x} \cap \overrightarrow{z} = \emptyset$$

$$P \downarrow_{\overline{x_1 \cdot \ldots \cdot x_n}} \;\; \textit{iff } P \equiv \nu \overrightarrow{z}((\overline{x_1 \cdot \ldots \cdot x_n}\langle \overrightarrow{y}\rangle.R + S) \mid Q) \textit{ where } \overrightarrow{x} \cap \overrightarrow{z} = \emptyset$$

We let $P \Downarrow_{x_1 \cdot \ldots \cdot x_n}$ iff $P \Rightarrow \downarrow_{x_1 \cdot \ldots \cdot x_n}$, and similarly for output barbs.

Definition 2.5 *The calculus $\mathrm{a}\pi_n$ is defined as the polyadic asynchronous π-calculus*

with polyadic synchronisation of degree n, that is

$$P ::= \mathbf{0} \mid P \mid Q \mid \nu x P \mid !P \mid x_1 \cdot \ldots \cdot x_n(\overrightarrow{y}).P \mid \overline{x_1 \cdot \ldots \cdot x_n}\langle \overrightarrow{y} \rangle$$

The only reduction axiom for $\mathrm{a}\pi_n$ *is*

$$x_1 \cdot \ldots \cdot x_n(y_1, \ldots, y_m).P \mid \overline{x_1 \cdot \ldots \cdot x_n}\langle z_1, \ldots, z_m \rangle \to P\{z_1, \ldots, z_m / y_1, \ldots, y_m\}$$

Note that $\mathrm{a}\pi_1$ is the standard asynchronous π-calculus. The *localised* π-calculus Lπ [2,7] is the subset of $\mathrm{a}\pi_1$ where the recipient of a name may only use it in output actions. We write $\mathrm{a}\pi_n^=$ to denote $\mathrm{a}\pi_n$ with matching $[x = y]P$.

Definition 2.6 *The calculus CCS_n is defined as the fragment of $\mathrm{f}\pi_n$ which has no name-passing and no matching or mismatching, that is*

$$P ::= P \mid Q \mid \nu x P \mid !P \mid \Sigma_i \alpha_i.P$$

where each α_i is of the form $x_1 \cdot \ldots \cdot x_n$ or $\overline{x_1 \cdot \ldots \cdot x_n}$.

The CCS_n synchronisation rule is

$$(\overline{x_1 \cdot \ldots \cdot x_n}.P + S) \mid (x_1 \cdot \ldots \cdot x_n.Q + T) \to P \mid Q \ .$$

Note that CCS_1 is a form of standard CCS. It resembles the CCS of [8] with replication instead of recursion.

Definition 2.7 *The calculus of* Mobile Ambients *(MA) [5] has the following syntax:*

$$P ::= \mathbf{0} \mid P \mid Q \mid \nu x P \mid !P \mid x[\,P\,] \mid \mathsf{in}\, x.P \mid \mathsf{out}\, x.P \mid \mathsf{open} x.P \mid \langle x \rangle \mid (x).P$$

Here $x[\,P\,]$ is an ambient named x enclosing P, and in, out, open are the *capabilities* for entering, leaving or dissolving ambients. We also have asynchronous, anonymous (no channel) output and input.[4] The free names $\mathsf{fn}(P)$ of a process P are defined much as for the π-calculus, with input and restriction being name-binding. Structural congruence and reduction rules are adapted from the π-calculus, with the following reduction axioms:

$$x[\,\mathsf{in}\, y.P \mid Q\,] \mid y[\,R\,] \to y[\,x[\,P \mid Q\,] \mid R\,]$$
$$y[\,x[\,\mathsf{out}\, y.P \mid Q\,] \mid R\,] \to x[\,P \mid Q\,] \mid y[\,R\,]$$
$$\mathsf{open} x.P \mid x[\,Q\,] \to P \mid Q$$
$$\langle x \rangle \mid (y).P \to P\{x/y\}$$

Barbs are defined by

$$P \downarrow_x \ \text{iff } P \equiv \nu \overrightarrow{z}(x[\,P\,] \mid Q) \text{ where } x \notin \overrightarrow{z} \ .$$

Pure public boxed MA (ppbMA) is got by omitting communication, restriction and the open capability. Recall that the open capability is omitted in the calculus of boxed ambients [3].

Let $\mathrm{MA}^{-\mathsf{in}}$ denote (full) MA with only the in capability omitted.

[4] Note that for simplicity we have just defined name-passing communication, whereas communication in [5] allows sequences of capabilities to be transmitted.

3 Weak Matching Systems

We present the weakened definition of matching system. Then we show that $a\pi_2$ has matching systems of every finite degree. We show that CCS_n does not have matching systems of degree $n+1$ or greater. We then show that matching systems are preserved by encodings satisfying certain properties. We deduce that there is no encoding from $a\pi_2$ to CCS satisfying those properties. We also present analogous results concerning ppbMA and CCS.

In matching systems [4], the idea is that clients C communicate with servers S and try to match their parameters, reporting success if there is a match. We change Carbone and Maffeis's definition of matching system to the following, which applies to all the calculi defined in Section 2:

Definition 3.1 *A* weak matching system (WMS) of degree n *is a tuple* $(C, S, x_1, \ldots, x_n)$ *where* C *and* S *are processes and* $x_1, \ldots, x_n$ *are distinct names, such that for all finite index sets* I *and* J*, and all injective substitutions* σ_i $(i \in I)$ *and* θ_j $(j \in J)$ *where* $\mathrm{dom}(\sigma_i) = \mathrm{dom}(\theta_j) = \{x_1, \ldots, x_n\}$*,*

$$(\prod_{i \in I} C\sigma_i \mid \prod_{j \in J} S\theta_j) \Downarrow_\omega \quad \textit{iff } \exists i \in I, j \in J \textit{ such that } \sigma_i = \theta_j \ .$$

Here ω *is a special name used for reporting a successful match. We require that* $\omega \notin \{x_1, \ldots, x_n\}$ *and that substitutions do not change any* x_i *into* ω*. Also substitutions should not map any* x_i *into a free name of* C *or* S*, other than* $x_1, \ldots, x_n$*. When convenient, we display parameters explicitly, writing* $C\sigma$ *as* $C\langle\sigma(x_1), \ldots, \sigma(x_n)\rangle$*.*

It is easy to see that, in a WMS $(C, S, \vec{x})$, all of $\vec{x}$ must be free in both C and S.

There are five changes from the pre-existing notion. Firstly, and most importantly, we do not require that if there is a match then *every* computation leads to success. Thus we have a "may" notion of success, rather than a "must" notion. Secondly, we do not use replication in the definition (for the server). Thirdly, we omit the identifier for the client, so that client and server are symmetrical. Fourthly, we allow both client and server to contain free names not drawn from $\vec{x}$. Fifthly, we require that parameters are distinct, so that we are dealing with permutations rather than substitutions in general. This last condition will be useful when we show that matching systems are preserved by encodings (Theorem 3.9).

Note that in standard process calculi a weak matching system never needs to use recursion or replication. It must be the case that $(C\langle\vec{x}\rangle \mid S\langle\vec{x}\rangle) \Downarrow_\omega$ by a finite computation. We can unfold recursion or replication enough to get this computation, and then set the recursion or replication part to the nil process $\mathbf{0}$. The modified client and server still give an ω barb when there is a match, and, since we have only reduced behaviour and not added any new behaviour, they still do not yield an ω barb when there is no match.

If a calculus has a WMS of degree n then it has WMSs of all smaller degrees:

Lemma 3.2 *Let* $m < n$*. If* $(C, S, x_1, \ldots, x_n)$ *is a WMS of degree* n *then* $(C, S, x_1, \ldots, x_m)$ *is a WMS of degree* m*.* □

We now show that $a\pi_1^=$ has weak matching systems of every degree:

Theorem 3.3 *For every* $n \geq 1$*,* $a\pi_1^=$ *has a WMS of degree* n*.*

Proof. (Sketch) We define C_n and S_n as follows:

$$C_n(x_1,\ldots,x_n) \stackrel{\text{df}}{=} x_1(z').(\textstyle\prod_{i=2}^{n} \overline{z'}\langle x_i\rangle)$$
$$S_n(x_1,\ldots,x_n) \stackrel{\text{df}}{=} \nu z(\overline{x_1}\langle z\rangle \mid S'_{n-1}\langle z, x_2,\ldots,x_n\rangle)$$
$$S'_{n-k}(z,x_{k+1},\ldots,x_n) \stackrel{\text{df}}{=} z(x'_{k+1}).([x_{k+1}=x'_{k+1}]S'_{n-k-1}\langle z, x_{k+2},\ldots,x_n\rangle)$$
$$\text{for } k = 1,\ldots,n-2$$
$$S'_1(z,x_n) \stackrel{\text{df}}{=} z(x'_n).([x_n=x'_n]\overline{\omega})$$

The server creates a new private name z, which is passed to the client on the first communication on x_1. The client then uses this private channel to send the other names back to the server. As each name is received, the server checks that it matches. Notice that the computation can fail even if conducted entirely between a matching client and server, due to the nondeterminism in the order in which the messages from the client are sent. This does not cause a problem, since we are dealing with weak matching systems—a single successful computation is enough. □

Remark 3.4 *We recall from [4] that matching can be encoded in* aπ_2*; the process* $[x=y]P$ *may be encoded as* $\nu z(\overline{z\cdot x} \mid z\cdot y.P)$ *where* z *is fresh. Hence Theorem 3.3 also holds for* aπ_2*. Note also that, in fact, the solution is written in* Lπ *with matching.*

We can also define matching systems using ambients:

Theorem 3.5 *For every* $n \geq 1$*, ppbMA has a WMS of degree* n*.*

Proof. Let

$$C_n \stackrel{\text{df}}{=} m[\,\mathsf{in}\,x_1.\cdots.\mathsf{in}\,x_n.\mathsf{out}\,x_n.\ldots.\mathsf{out}\,x_1.\omega[\,\mathsf{out}\,m\,]\,]$$
$$S_n \stackrel{\text{df}}{=} x_1[\,x_2[\ldots x_n[\,]\ldots]\,]\ .$$

The idea is that the client enters successively the stacked $x_1,\ldots,x_n$ ambients of the server, before returning to the top level to report success. The client simply gets stuck if there is no match. □

We next investigate matching systems for CCS.

Theorem 3.6 *Let* $m,n \geq 1$*. Then* CCS_n *has a WMS of degree* m *if and only if* $n \geq m$*.*

Proof. (Sketch) First suppose that $n \geq m$. We define a WMS of degree m in CCS_n as follows:

$$C_m \stackrel{\text{df}}{=} \overline{x_1\cdot\ldots\cdot x_m} \qquad S_m \stackrel{\text{df}}{=} x_1\cdot\ldots\cdot x_m.\overline{\omega}$$

Notice that this WMS is guaranteed to succeed, and so it is in fact a *strong* MS, to be defined in Section 4.

For the converse direction, by Lemma 3.2 it is enough to show that CCS_n does not have a WMS of degree $n+1$. So suppose for a contradiction that $(C,S,x_1,\ldots,x_{n+1})$ is a WMS of degree $n+1$ in CCS_n. We shall show that there is a combination of clients and servers which does not contain a match, and yet erroneously returns success.

There is $k \geq 0$ and there are C_i, S_i $(0 \leq i \leq k)$ such that

$$C \mid S = C_0 \mid S_0 \to \cdots \to C_k \mid S_k \quad \text{where } (C_k \mid S_k) \downarrow_\omega \ .$$

This holds because there is a match between the single client and and the single server (using the identity substitution in both cases). Note that during the computation we may have to extrude the scope of restrictions in order to obtain the necessary redex, but we can then immediately return the scopes so that they lie entirely within C_{i+1} or S_{i+1}. This returning of scopes would not in general be possible in the π-calculus, where restricted names can be transmitted along channels, resulting in more than one process sharing the same restricted name.

Let $x'_1, \ldots, x'_{n+1}$ be distinct fresh names different from $x_1, \ldots, x_{n+1}$. Let $s = s_1 \cdots s_{n+1}$ range over binary strings in $\{0,1\}^{n+1}$. Let σ_s be the substitution which sets

$$\sigma_s(x_i) \stackrel{\text{df}}{=} \begin{cases} x_i & \text{if } s_i = 0 \\ x'_i & \text{if } s_i = 1 \end{cases}$$

Let $E = \{s \in \{0,1\}^{n+1} : s \text{ has an even number of 1s}\}$ and $O = \{s \in \{0,1\}^{n+1} : s \text{ has an odd number of 1s}\}$. For $i = 0, \ldots, k$ let

$$P_i \stackrel{\text{df}}{=} \prod_{s \in E} C_i\sigma_s \mid \prod_{s \in O} S_i\sigma_s$$

Then P_0 does not contain a match. We show that $P_i \to^{2^n} P_{i+1}$ for $i = 0, \ldots, k-1$. Since plainly $P_k \downarrow_\omega$, we shall have a contradiction.

There are various cases

(i) Suppose that $C_i \to C_{i+1}$ with $S_{i+1} = S_i$. Then for each $s \in E$ we have $C_i\sigma_s \to C_{i+1}\sigma_s$, and for each $s \in O$ we have $S_{i+1}\sigma_s = S_i\sigma_s$.

(ii) The case where $S_i \to S_{i+1}$ with $C_{i+1} = C_i$ is handled like the preceding case.

(iii) Suppose that $C_i \mid S_i \to C_{i+1} \mid S_{i+1}$ by a communication on channel $y_1 \cdot \ldots \cdot y_n$. Let j be such that $x_j \notin \overrightarrow{y}$. Let $s \in E$. Let t be the same as s except that $t_j = 1 - s_j$. Then $t \in O$. Also, $C_i\sigma_s$ and $S_i\sigma_t$ can communicate on channel $\sigma_s(y_1) \cdot \ldots \cdot \sigma_s(y_n)$ to produce $C_{i+1}\sigma_s \mid S_{i+1}\sigma_t$. In this way we pair off all clients and servers and we produce P_{i+1} after 2^n reductions.

□

The next result suggests that the in capability is needed to obtain WMSs for MA.

Theorem 3.7 *For $n \geq 2$, there is no WMS of degree n in pure $MA^{-\mathsf{in}}$.*

Proof. (Sketch) We adapt the method used in the proof of Theorem 3.6. We suppose that we have a WMS of degree 2, and show for a contradiction that

$$C\langle x_1, x_2\rangle \mid C\langle x'_1, x'_2\rangle \mid S\langle x'_1, x_2\rangle \mid S\langle x_1, x'_2\rangle$$

has a successful computation. This is possible because the clients and servers can only interact at the top level of the ambient tree, due to the absence of the in capability. □

Conjecture 3.8 *For $n \geq 2$ there is no WMS of degree n in pure MA without the* out *capability.* □

We now establish conditions under which matching systems are preserved when encoding one language in another. Our result (Theorem 3.9) will apply to all the languages defined in Section 2.

We assume that we are dealing with process calculi L which satisfy: (1) for any permutation σ, $P \Downarrow_\omega$ iff $\sigma(P) \Downarrow_{\sigma(\omega)}$; (2) $\mathcal{N}$ is infinite. These conditions are satisfied by any process calculus in the π-calculus family (including ambient calculi).

The next theorem shows that weak matching systems are preserved by encodings satisfying certain conditions. The first two conditions are similar to those used by Palamidessi [9]. In the third condition, the injection φ and its properties are similar to Gorla's "strict renaming policy" [6]. The idea is that names of the source language are mapped across to unique names in the target language by φ, with the names which are not in the range of φ being available as "reserved names" for use in the encoding (so φ could be the identity if the encoding required no reserved names). The encoding of a process P should not depend on the particular names in P, since names have no structure or meaning. This idea is expressed by requiring a property of invariance under injective substitution, mediated by φ.

Theorem 3.9 *Let L and L' be process calculi. Let $[\![-]\!] : L \to L'$ be an encoding satisfying:*

(i) *$P \Downarrow_\omega$ iff $[\![P]\!] \Downarrow_\omega$;*

(ii) *$[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$;*

(iii) *There is an injective $\varphi : \mathcal{N} \to \mathcal{N}$ with $\varphi(\omega) = \omega$, such that for all finite injective substitutions σ, if P is such that $\mathrm{rge}(\sigma) \cap \mathsf{fn}(P) = \emptyset$ then we have $[\![P\sigma]\!] = [\![P]\!]\sigma'$, where the injective substitution σ' is defined by*

$$\begin{array}{ll} \sigma'(\varphi(x)) = \varphi(\sigma(x)) & \text{if } x \in \mathrm{dom}(\sigma) \\ \sigma'(x) \text{ undefined} & \text{otherwise} \end{array}$$

Let $(C, S, \vec{x})$ be a weak matching system of degree n in L. Then $([\![C]\!], [\![S]\!], \overrightarrow{\varphi(x)})$ is a weak matching system of degree n in L'.

Proof. (Sketch) Consider

$$P \stackrel{\mathrm{df}}{=} \prod_{i \in I} [\![C]\!]\sigma_i \mid \prod_{j \in J} [\![S]\!]\theta_j$$

where $\mathrm{dom}(\sigma_i) = \mathrm{dom}(\theta_j) = \{\varphi(x_1), \ldots, \varphi(x_n)\}$. We need to show that $P \Downarrow_\omega$ iff P has a match, i.e. there are $i \in I$ and $j \in J$ such that $\sigma_i = \theta_j$.

Let $A = \bigcup_{i \in I} \mathrm{rge}(\sigma_i) \cup \bigcup_{j \in J} \mathrm{rge}(\theta_j)$. Let B be a set of names in bijection with A via $f : A \to B$, such that for each $x \in B$, both x and $\varphi(x)$ are fresh. This is always possible, since we assume that $\mathcal{N}$ is infinite.

For each $i \in I$ and each $k = 1, \ldots, n$, let

$$\sigma'_i(x_k) \stackrel{\mathrm{df}}{=} f(\sigma_i(\varphi(x_k))) \ .$$

Similarly, for each $j \in J$ and each $k = 1, \ldots, n$, let

$$\theta'_j(x_k) \stackrel{\mathrm{df}}{=} f(\theta_j(\varphi(x_k))) \ .$$

Then σ'_i, θ'_j are finite injective substitutions. Also $\sigma'_i(x_k)$, $\theta'_j(x_k) \notin \{\omega\} \cup \mathsf{fn}(P) \cup \mathsf{fn}(Q)$, since all $x \in B$ are fresh.

Now P has a match iff

$$Q \stackrel{\text{df}}{=} \prod_{i \in I} C\sigma'_i \mid \prod_{j \in J} S\theta'_j$$

has a match. This is because Q has a match iff $\exists i, j. \forall k.\, f(\sigma_i(\varphi(x_k))) = f(\theta_j(\varphi(x_k)))$ iff $\exists i, j. \forall k.\, \sigma_i(\varphi(x_k)) = \theta_j(\varphi(x_k))$ (since f is a bijection) iff P has a match.

By property (iii) of the encoding, for each $i \in I$ there is σ''_i such that $[\![C\sigma'_i]\!] = [\![C]\!]\sigma''_i$ where $\sigma''_i(\varphi(x_k)) = \varphi(\sigma'_i(x_k))$. Similarly, for each $j \in J$ there is θ''_j such that $[\![S\theta'_j]\!] = [\![S]\!]\theta''_j$ where $\theta''_j(\varphi(x_k)) = \varphi(\theta'_j(x_k))$.

Now Q has a match iff $Q \Downarrow_\omega$ (since $(C, S, \overrightarrow{x})$ is a WMS). Also, $Q \Downarrow_\omega$ iff $[\![Q]\!] \Downarrow_\omega$ (property (i) of the encoding). Using property (ii) of the encoding, we have $[\![Q]\!] = R$, where

$$R \stackrel{\text{df}}{=} \prod_{i \in I} [\![C]\!]\sigma''_i \mid \prod_{j \in J} [\![S]\!]\theta''_j$$

Notice that $\sigma''_i(\varphi(x_k)) = \varphi(\sigma'_i(x_k)) = \varphi(f(\sigma_i(\varphi(x_k))))$. Similarly, $\theta''_j(\varphi(x_k)) = \varphi(\theta'_j(x_k)) = \varphi(f(\theta_j(\varphi(x_k))))$. Also note that $\mathrm{rge}(\sigma'') \cap ((\mathsf{fn}([\![C]\!]) \cup \mathsf{fn}([\![S]\!])) \setminus \{\varphi(x_1), \ldots, \varphi(x_n)\}) = \emptyset$, since all $y \in \varphi(B)$ are fresh. Similarly for $\mathrm{rge}(\theta'')$. Since f and φ are injective, we can extend their composition $\varphi(f(\cdot))$ to a suitable permutation ρ which leaves ω unchanged, such that $R = P\rho$. But then $R \Downarrow_\omega$ iff $P \Downarrow_\omega$ (property of L').

Combining, we have: P has a match iff $P \Downarrow_\omega$, as required. □

We can use Theorem 3.9 and our various preceding positive and negative results to state some non-encodability results:

Theorem 3.10 *There is no encoding satisfying the conditions of Theorem 3.9 from* $\mathrm{a}\pi_1^=$ *to* CCS_n *(all* $n \geq 1$*).*

Proof. By Theorems 3.3, 3.6 and 3.9. □

In connection with Theorem 3.10, we note that Carbone and Maffeis showed that there is no sensible encoding from $\mathrm{a}\pi_1^=$ into the standard π-calculus with mixed choice (and without matching) [4, Theorem 4.1]. Our result here uses different conditions and holds for all levels of polyadic synchronisation in CCS.

Theorem 3.11 *There is no encoding satisfying the conditions of Theorem 3.9 from ppbMA to* CCS_n *(all* $n \geq 1$*).*

Proof. By Theorems 3.5, 3.6 and 3.9. □

Concerning Theorem 3.11, Phillips and Vigliotti [11] showed there is no encoding (under different conditions) from pure public MA (with open) to CCS. Previously they showed that there is no encoding (under yet other conditions) from pure public boxed MA to $\mathrm{a}\pi_1$ [10].

4 Strong Matching Systems

In this section we investigate *strong* matching systems, where if there is a match then every computation is *guaranteed* to succeed. We show that the full π-calculus $\mathrm{f}\pi_1$ has strong matching systems of every finite degree (Theorem 4.4).

Definition 4.1 *A* strong matching system (SMS) of degree n *is a tuple* $(C, S, x_1, \ldots, x_n)$ *where* C *and* S *are processes and* $x_1, \ldots, x_n$ *are distinct names, such that for all finite index sets* I *and* J*, and all substitutions* σ_i $(i \in I)$ *and* θ_j $(j \in J)$ *where* $\mathrm{dom}(\sigma_i) = \mathrm{dom}(\theta_j) = \{x_1, \ldots, x_n\}$*, defining*

$$MS \stackrel{\mathrm{df}}{=} \prod_{i \in I} C\sigma_i \mid \prod_{j \in J} S\theta_j$$

(i) *if* $MS \Downarrow_\omega$ *then* $\exists i \in I, j \in J.\ \sigma_i = \theta_j$*;*

(ii) *if* $\exists i \in I, j \in J.\ \sigma_i = \theta_j$ *then* $\forall MS'.\ MS \Rightarrow MS'$ *implies* $MS' \Downarrow_\omega$*;*

(iii) *there are no infinite reduction sequences starting from* MS*.*

A replicated SMS *(!SMS for short) is defined as an SMS, except that we require the servers to be replicated, so that*

$$MS \stackrel{\mathrm{df}}{=} \prod_{i \in I} C\sigma_i \mid \prod_{j \in J} !S\theta_j$$

Observe that if $(C, S, \overrightarrow{x})$ is a !SMS then $(C, !S, \overrightarrow{x})$ is an SMS. Also, if $(C, S, \overrightarrow{x})$ is an SMS then $(C, S, \overrightarrow{x})$ is a WMS.

The notion of !SMS is quite close to the original formulation of matching system in [4]. It differs from the original MS in two ways: Firstly, we omit the identifier for the client, so that client and server are symmetrical. Secondly, we allow both client and server to contain free names not drawn from $\overrightarrow{x}$.

The next result is similar to [4, Theorem 4.2]:

Theorem 4.2 *For all non-negative integer numbers* n *and* m*, there is a* !*SMS of degree* m *in* $\mathrm{f}\pi_n$ *if and only if* $n \geq m$*.*

Proof. ($\Leftarrow$) Choosing

$$C_m \stackrel{\mathrm{df}}{=} \overline{x_1 \cdot \ldots \cdot x_m}\langle\rangle \qquad S_m \stackrel{\mathrm{df}}{=} x_1 \cdot \ldots \cdot x_m(z).\overline{\omega}\langle\rangle$$

we have that $(C_m, S_m, x_1, \ldots, x_m)$ is a strong matching system of degree m.

($\Rightarrow$) The idea is that a client can be endlessly "fooled" into interaction with servers which only partially match, giving rise to an infinite computation.

Consider the minimal case where $m = n + 1$ and suppose $(C, S, x_1, \ldots, x_m)$ is a !SMS of degree m in π_n. Let σ be an injective substitution of fresh names. Then

$$P \stackrel{\mathrm{df}}{=} C\sigma \mid S\sigma$$

is a matching instance of $(C, S, x_1, \ldots, x_m)$. Note that for any such σ and any R, if there is a Q such that $R\sigma \Rightarrow Q$ then there is also an R' such that $R \Rightarrow R'$ and $Q = R'\sigma$.

By point (iii) of Definition 4.1, there must be C' and S' such that $C\sigma \Rightarrow C'\sigma$ and $C'\sigma \not\rightarrow$, and similarly $S\sigma \Rightarrow S'\sigma$ and $S'\sigma \not\rightarrow$. By point (ii) of Definition 4.1, it must be the case that $P \Rightarrow (C'\sigma \mid S'\sigma) \Downarrow_\omega$. By the contrapositive of point (i) of Definition 4.1, it must be the case that $C'\sigma \not\downarrow_\omega$ and $S\sigma \not\downarrow_\omega$. Hence, it must be the case that $C'\sigma \mid S'\sigma \rightarrow P' \Downarrow_\omega$ for some appropriate P'.

By definition of reduction, without loss of generality, we can assume that $C'\sigma \downarrow_{\overrightarrow{a_1}}, \ldots, C'\sigma \downarrow_{\overrightarrow{a_k}}$ and $S'\sigma \downarrow_{\overline{\overrightarrow{a_1}}}, \ldots, S'\sigma \downarrow_{\overline{\overrightarrow{a_k}}}$, where $\overrightarrow{a_1}, \ldots, \overrightarrow{a_k}$ are all of the possible channels on which the two processes are ready to communicate. Since $m > n$, for

each $j \in [1..k]$ there exists i such that $\sigma(x_i) \notin \overrightarrow{a_j}$. Let ρ_j be defined as $\rho_j(x_i) = d_j$, for a fresh d_j, and $\rho_j(x_h) = \sigma(x_h)$ otherwise. By construction, since both σ and each ρ_j are injective and fresh, we have $S\rho_j \Rightarrow S'\rho_j$. Since σ and ρ_j agree on $\overrightarrow{a_j}$, it must be the case that $S'\rho_j \downarrow_{\overline{\overrightarrow{a_j}}}$. By the contrapositive of point (i) of Definition 4.1, it must be the case that

$$P_0^j \stackrel{\text{df}}{=} (C\sigma \mid S\rho_j) \not\Downarrow_\omega .$$

However, because of the complementary barbs, there must be P_1^j such that $P_0^j \rightarrow P_1^j$. Still, since

$$P_2^j \stackrel{\text{df}}{=} P_0^j \mid S\sigma$$

is a valid instance of a matching system, by point (ii) it must be the case that $P_2^j \Downarrow_\omega$. Moreover, since

$$P_2^j \Rightarrow P_3^j \stackrel{\text{df}}{=} (P_1^j \mid S\sigma)$$

it must be the case that $P_3^j \Downarrow_\omega$.

We have established above that there must be S' such that $S\sigma \Rightarrow S'\sigma$ and $S'\sigma \not\rightarrow$, and $S'\sigma \downarrow_{\overline{\overrightarrow{a_1}}}, \ldots, S'\sigma \downarrow_{\overline{\overrightarrow{a_k}}}$. Similarly, there must be P_4^j such that $P_1^j \Rightarrow P_4^j \downarrow_{\overline{a_i}}$ for some i in $[1..k]$. By considering now

$$P_0 \stackrel{\text{df}}{=} C\sigma \mid \prod_{j \in [1..k]} !S\rho_j$$

we have a contradiction because the system can enter a loop: each $S\rho_j$ intercepts the corresponding attempt that $C\sigma$ must keep repeating in order to communicate with a potential matching server $S\sigma$. □

Theorem 4.3 *For $n \geq 2$, there is no !SMS of degree n in MA.*

Proof. The idea is similar to the proof of Theorem 4.2. □

By contrast with Theorem 4.2, we show that $\mathrm{f}\pi_1$ is strong enough to have SMSs of all degrees:

Theorem 4.4 *There is a strong matching system of degree n in $\mathrm{f}\pi_1$.*

Proof. (Sketch) Lists and operations on lists can be encoded in $\mathrm{f}\pi_1$ without introducing divergence, using only restricted names (we consider the encoding given by Turner [13]). For example the list $[a, b]$, accessible through channel x, is represented by the process $\nu y, z(!x(n, c).\overline{c}\langle a, y\rangle \mid !y(n, c).\overline{c}\langle b, z\rangle \mid !z(n, c).\overline{n}\langle\rangle)$. Note that this list above can be passed around as a single value by passing the name x. Below, we use the context $L[-] \stackrel{\text{df}}{=} \nu \overrightarrow{l}\,(L \mid -)$ to denote the machinery to implement lists and head, tail, concatenation, etc. operations in $\mathrm{f}\pi_1$. We assume that the names $\overrightarrow{l}$, used to implement the list operations, are fresh, and that $\mathsf{fn}(L[\mathbf{0}]) = \emptyset$. Consider the processes

$$C_n \stackrel{\text{df}}{=} L[\overline{e}\langle[x_1, \ldots, x_n], [\,]\rangle]$$

$$S_m \stackrel{\text{df}}{=} L[\nu a \begin{pmatrix} \overline{a}\langle[\,], [x_1, \ldots, x_n]\rangle \mid !a(x, y).(\overline{e}\langle x, y\rangle + \\ e(z, w).F(x@z, y@w, x', y').\overline{a}\langle x', y'\rangle) \end{pmatrix}]$$

where function F takes as input the lists $x@z, y@w$ and returns the lists x', y' obtained by removing all the matching pairs from $x@z$ and $y@w$ (@ stands for list concatenation). For each matching pair, F produces the barb ω. Such a function can be implemented in $f\pi_1$ without introducing divergence. □

There is an essential use of mixed choice in the proof of Theorem 4.4. We conjecture that without mixed choice it is impossible to get SMSs of degree higher than the level of synchronisation in the language:

Conjecture 4.5 *For $m > n$ there is no SMS of degree m in $\mathrm{a}\pi_n$.*

5 Conclusions and Further Work

We have adapted the notion of matching system from earlier work by Carbone and Maffeis. We have seen that there are two main types of matching system, the weak and the strong, depending on whether successful termination is possible or guaranteed (in the event of a match between some client and some server). In the strong case, there are two subtypes of matching system, depending on whether the server is required to be a replication or not (the former being the stronger of the two).

These notions can be used to "grade" process calculi according to how good they are at treating synchronisation on several different names as a single transaction.

We have seen that the full π-calculus is strong enough to have strong matching systems of all degrees, but not strong enough to have replicated strong matching systems of degree greater than one.

We also showed that the asynchronous π-calculus with matching has weak matching systems of every finite degree. Our work leaves open the question of whether the asynchronous π-calculus has a strong matching system of degree two or higher. We conjecture that the answer is no.

We showed that the calculus of Mobile Ambients has weak matching systems of all finite degrees. Furthermore, MA does not have replicated strong matching systems of degree two or higher. Our work leaves open the question of whether MA has a strong matching system of degree two or higher. Again, we conjecture that the answer is no.

We showed that CCS does not have weak matching systems of degree greater than one. By our result on preservation of weak matching systems by suitable encodings, we could deduce a non-encodability result for the asynchronous π-calculus with matching into CCS with all levels of polyadic synchronisation.

Acknowledgements

We thank Uwe Nestmann for a helpful discussion, and the referees for their helpful suggestions.

References

[1] Banach, R. and F. van Breugel, *Mobility and modularity: expressing pi-calculus in CCS (extended abstract)* (1998), draft.

[2] Boreale, M., *On the expressiveness of internal mobility in name-passing calculi*, Theoretical Computer Science **195** (1998), pp. 205–226.

[3] Bugliesi, M., G. Castagna and S. Crafa, *Access control for mobile agents: the calculus of Boxed Ambients*, ACM Transactions on Programming Languages and Systems **26** (2004), pp. 57–124.

[4] Carbone, M. and S. Maffeis, *On the expressive power of polyadic synchronisation in π-calculus*, Nordic Journal of Computing **10** (2003), pp. 70–98.

[5] Cardelli, L. and A.D. Gordon, *Mobile ambients*, Theoretical Computer Science **240** (2000), pp. 177–213.

[6] Gorla, D., *Comparing calculi for mobility via their relative expressive power*, Technical Report 05/2006, Dip. di Informatica, Univ. di Roma "La Sapienza", Italy (2006).

[7] Merro, M. and D. Sangiorgi, *On asynchrony in name-passing calculi*, Mathematical Structures in Computer Science **14** (2004), pp. 715–767.

[8] Milner, R., "Communicating and Mobile Systems: the π-calculus," Cambridge University Press, 1999.

[9] Palamidessi, C., *Comparing the expressive power of the synchronous and the asynchronous π-calculi*, Mathematical Structures in Computer Science **13** (2003), pp. 685–719.

[10] Phillips, I.C.C. and M. Vigliotti, *Electoral systems in ambient calculi*, in: *Proceedings of 7th International Conference on Foundations of Software Science and Computation Structures, FoSSaCS 2004*, Lecture Notes in Computer Science **2987** (2004), pp. 408–422.

[11] Phillips, I.C.C. and M. Vigliotti, *Leader election in rings of ambient processes*, Theoretical Computer Science **356** (2006), pp. 468–494.

[12] Sangiorgi, D., *π-calculus, internal mobility and agent-passing calculi*, Theoretical Computer Science **167** (1996), pp. 235–274.

[13] Turner, D.N., "The Polymorphic Pi-calculus: Theory and Implementation," Ph.D. thesis, University of Edinburgh (1995).

Linearity, Persistence and Testing Semantics in the Asynchronous Pi-Calculus

Diletta Cacciagrano[2], Flavio Corradini[3]

Dipartimento di Matematica e Informatica
Università degli Studi di Camerino, Italy

Jesús Aranda [1,4]

INRIA Futurs, LIX École Polytechnique, France
Escuela de Ingeniería de Sistemas y Computación, Universidad del Valle, Colombia

Frank D. Valencia[5]

CNRS and LIX École Polytechnique, France

Abstract

In [24] the authors studied the expressiveness of persistence in the asynchronous π-calculus (Aπ) wrt weak barbed congruence. The study is incomplete because it ignores the issue of divergence. In this paper, we present an expressiveness study of persistence in the asynchronous π-calculus (Aπ) wrt De Nicola and Hennessy's testing scenario which is sensitive to divergence. Following [24], we consider Aπ and three sub-languages of it, each capturing one source of persistence: the *persistent-input* calculus (PIAπ), the *persistent-output* calculus (POAπ) and *persistent* calculus (PAπ). In [24] the authors showed encodings from Aπ into the semi-persistent calculi (i.e., POAπ and PIAπ) correct wrt weak barbed congruence. In this paper we prove that, under some general conditions, there cannot be an encoding from Aπ into a (semi)-persistent calculus preserving the *must* testing semantics.

Keywords: Asynchronous Pi-Calculus, Linearity, Persistence, Testing Semantics.

1 Introduction

In [24] the authors present an expressiveness study of linearity and persistence of processes. Since several calculi presuppose persistence on their processes, the authors address the expressiveness issue of whether such persistence restricts the

[1] The work of Jesús Aranda has been supported by COLCIENCIAS (Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología "Francisco José de Caldas") and INRIA Futurs.

[2] Email:diletta.cacciagrano@unicam.it

[3] Email:flavio.corradini@unicam.it

[4] Email:jesus.aranda@lix.polytechnique.fr

[5] Email:frank.valencia@lix.polytechnique.fr

This is a preliminary version. The final version will be electronically published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.com/locate/entcs

systems that we can specify, model or reason about in the framework. Their work is conducted using the standard notion weak barbed congruence and hence it ignores divergence issues. Since divergence plays an important role in expressiveness studies, particularly in those studies involving persistence, in this work we aim at extending and strengthening their study by using the standard notion of testing equivalences. As elaborated below, our technical results contrast and complement those in [24]. More importantly, our results also clarify and support informal expressiveness claims in the literature.

Motivation: *Linearity* is present in process calculi such as CCS, CSP, the π-calculus [20] and Linear CCP [31,14] where messages are consumed upon being received. In the π-calculus the system $\bar{x}z|x(y).P\ |x(y).Q$ represents a message with a datum z, tagged with x, that can be *consumed* by either $x(y).P$ or $x(y).Q$. *Persistence of messages* is present in several process calculi. Perhaps the most prominent representative of such calculi is Concurrent Constraint Programming (CCP) [32]. Here the messages (or items of information) can be read but, unlike in Linear CCP, they cannot be consumed. Other prominent examples can be found in the context of calculi for analyzing and describing security protocols: Crazzolara and Winskel's SPL [12], the Spi Calculus variants by Fiore and Abadi [15] and by Amadio et all [2], and the calculus of Boreale and Buscemi [5] are operationally defined in terms of configurations containing messages which cannot be consumed. *Persistent receivers* arise, e.g. in the notion of *omega receptiveness* [29] where the input of a name is always available—but always with the same continuation. In the π-calculus persistent receivers are used, for instance, to model functions, objects, higher-order communications, or procedure definitions. Furthermore, persistence of *both* messages and receivers arise in the context of CCP with universally-quantified persistent ask operations. In the context of calculi for security, persistent receivers can be used to specify protocols where principals are willing to run an unbounded number of times (and persistent messages to model the fact that every message can be remembered by the spy). In fact, the approach of specifying protocols in a persistent setting, with an unbounded number of sessions, has been explored in [4] by using a classic logic Horn clause representation of protocols (rather than a linear logic one).

Expressiveness of Persistence - Drawbacks and Conjectures: The study in [24] is conducted in the *asynchronous* π-calculus (Aπ), which naturally captures the persistent features mentioned above. Persistent messages (and receivers) can simply be specified using the *replication* operator of the calculus which creates an unbounded number of copies of a given process. In particular, the authors in [24] investigate the existence of encodings from Aπ into three sub-languages of it, each capturing one source of persistence: the *persistent-input* calculus (PIAπ), defined as Aπ where inputs are replicated; *persistent-output* calculus (POAπ), defined dually, i.e. outputs rather than inputs are replicated; *persistent* calculus (PAπ), defined as Aπ but with all inputs and outputs replicated. The main result basically states that we need one source of linearity, i.e. either on inputs (PIAπ) or outputs (POAπ) to encode the behavior of arbitrary Aπ processes via weak barbed congruence. Nevertheless, the main drawback of the work [24] is that the notion of correct

encoding is based on weak barbed bisimulation (congruence), which is not sensitive to *divergence*. In particular, the encoding provided in [24] from Aπ into PIAπ is weak barbed congruent preserving but not divergence preserving. Although in some situations divergence may be ignored, in general it is an important issue to consider in the correctness of encodings [8,17,16,18,7].

In fact, the informal claims of extra expressivity of Linear CCP over CCP in [3,14] are based on discrimination introduced by divergence that is clearly ignored by the standard notion of weak bisimulation. Furthermore, the author of [11] suggests as future work to extend SPL, which uses only persistent messages and replication, with recursive definitions to be able to program and model recursive protocols such as those in [1,25]. Nevertheless, one can give an encoding of recursion in SPL from an easy adaptation of the composition between the Aπ encoding of recursion [30] (where recursive calls are translated into *linear* Aπ outputs and recursive definitions into persistent inputs) and the encoding of Aπ into POAπ in [24]. The resulting encoding is correct up-to weak bisimulation. The encoding of Aπ into POAπ, however, introduces divergence and hence the composite encoding does not seem to invalidate the justification for extending SPL with recursive definitions. The above works suggest that the expressiveness study of persistence is relevant but incomplete if divergence is not taken into account.

This work: In this paper we shall therefore study the existence of encodings from Aπ into the persistent sub-languages mentioned above using testing semantics [13].

Our main contribution is to provide a uniform and general result stating that under some reasonable conditions Aπ cannot be encoded into any of the above (semi-) persistent calculi while preserving the *must* testing semantics. The general conditions involve compositionality on the encoding of constructors such as parallel composition, prefix, and replication. The main result contrasts and completes the ones in [24]. It also supports the informal claims of extra expressivity mentioned above. We shall also state other more specialized impossibility results for must preserving encodings from Aπ into the semi-persistent calculi, focusing on specific properties of each target calculus. This helps clarifying some previous assumptions on the interplay between syntax and semantics in encodings of process calculi. We believe that, since the study is conducted in Aπ with well-established notions of equivalence, we can easily adapt our results to other asynchronous frameworks such as CCP languages and the above-mentioned calculi for security.

2 The Calculi

Here we define the calculi we study. We first recall the (monadic) *asynchronous π-calculus* (Aπ). The other calculi are defined as syntactic restrictions of Aπ.

2.1 The asynchronous pi-calculus

Let $\mathcal{N}$ (ranged over by $x, y, z, \ldots$) be a set of names. The set of the asynchronous π-calculus processes (ranged over by P, Q, $R \ldots$) is generated by the following grammar:

$$P, Q, \ldots ::= \quad 0 \;\Big|\; \bar{x}z \;\Big|\; x(y).P \;\Big|\; P \mid Q \;\Big|\; (\nu x)P \;\Big|\; !\,P$$

Intuitively, an *output* $\bar{x}z$ represents a message z tagged with a name x indicating that it can be received (or *consumed*) by an *input process* $x(y).P$ which behaves, upon receiving z, as $P\{z/y\}$. Furthermore, $x(y).P$ binds the names y in P. The other binder is the *restriction* $(\nu x)P$ which declares a name x private to P. The *parallel composition* $P \mid Q$ means P and Q running in parallel. The *replication* $!P$ means $P \mid P \mid \ldots$, i.e., $!P$ represents a *persistent resource*.

We use the standard notations $bn(Q)$ for the *bound names* in Q, and $fn(Q)$ for the *free names* in Q. The set of names of P is defined as $n(P) = fn(P) \cup bn(P)$. We write $(\nu x_1 \ldots x_n)P$ to denote $(\nu x_1)\ldots(\nu x_n)P$. We let $\sigma, \vartheta \ldots$ range over (non-capturing) substitutions of names on processes.

The *reduction* relation $\longrightarrow$ is the least binary relation on processes satisfying the rules in Table 1. $\xrightarrow{*}$ denotes the reflexive, transitive closure of $\longrightarrow$. The reductions are quotiented by the *structural congruence* relation $\equiv$.

Definition 2.1 [Structural equivalence] Let $\equiv$ be the smallest congruence over processes satisfying α-equivalence, the commutative monoid laws for composition with 0 as identity, the replication law $!P \equiv P \mid !P$, the restriction laws $(\nu x)0 \equiv 0$, $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$ and the extrusion law: $(\nu x)(P \mid Q) \equiv P \mid (\nu x)Q$ if $x \notin fn(P)$.

$$\text{Com} \quad \bar{x}z \mid x(y).P \longrightarrow P\{z/y\}$$

$$\text{Par} \quad \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad \text{Res} \quad \frac{P \longrightarrow P'}{(\nu x)P \longrightarrow (\nu x)P'}$$

$$\text{Cong} \quad \frac{P \equiv P' \quad P' \longrightarrow Q' \quad Q' \equiv Q}{P \longrightarrow Q}$$

Table 1
Reduction Rules.

2.2 The (semi-)persistent calculi

The *persistent-input* calculus PIAπ results from Aπ by requiring all input processes to be replicated. Processes in PIAπ are generated by the following grammar:

$$P, Q, \ldots ::= \quad 0 \;\Big|\; !\,x(y).P \;\Big|\; \bar{x}y \;\Big|\; P \mid Q \;\Big|\; (\nu x)P \;\Big|\; !\,P$$

The *persistent-output* calculus POAπ arises as from Aπ by requiring all outputs to be replicated. Processes in POAπ are generated by the following grammar:

$$P, Q, \ldots ::= \quad 0 \;\Big|\; x(y).P \;\Big|\; !\,\bar{x}y \;\Big|\; P \mid Q \;\Big|\; (\nu x)P \;\Big|\; !\,P$$

Finally, we have the *persistent* calculus PAπ, a subset of Aπ where output and input processes must be replicated. Processes in PAπ are generated by the following grammar:

$$P, Q, \ldots ::= \quad 0 \;\Big|\; !\,x(y).P \;\Big|\; !\,\bar{x}y \;\Big|\; P \mid Q \;\Big|\; (\nu x)P \;\Big|\; !\,P$$

The relation $\longrightarrow$ for PIAπ, POAπ and PAπ can be equivalently defined as in Table 1, with Com replaced respectively with Com(PIAπ), Com(POAπ) and Com(PAπ) rules (Table 2). The new rules reflect the *persistent-input* and *linear-output* nature of PIAπ (Rule Com(PIAπ)), the *linear-input* and *persistent-output* nature of POAπ (Rule Com(POAπ)), and the *persistent* nature of PAπ (Rule Com(PAπ)).

Com(PIAπ)	$\bar{x}z \mid !\,x(y).P \longrightarrow P\{z/y\} \mid !\,x(y).P$
Com(POAπ)	$!\,\bar{x}z \mid x(y).P \longrightarrow !\,\bar{x}z \mid P\{z/y\}$
Com(PAπ)	$!\,\bar{x}z \mid !\,x(y).P \longrightarrow P\{z/y\} \mid !\,\bar{x}z \mid !\,x(y).P$

Table 2
Reduction Rules

Notation 2.1 *We shall use $\mathcal{P}$ to range over the set of the calculi so-far defined $\{A\pi, PIA\pi, POA\pi, PA\pi\}$.*

3 Testing Semantics

In [13] De Nicola and Hennessy propose a framework for defining pre-orders that is widely acknowledged as a realistic scenario for system testing. It means to define formally when one process is a correct implementation of another considering specially unsafe contexts, in which is particularly important what is the revealed information of the process in any context or test. In this section we summarize the basic definitions behind the testing machinery for the π-calculi.

Definition 3.1 [Observers]

- The set of names $\mathcal{N}$ is extended as $\mathcal{N}' = \mathcal{N} \cup \{\omega\}$ with $\omega \notin \mathcal{N}$. By convention we let $fn(\omega) = \{\omega\}$ and $bn(\omega) = \emptyset$ (ω is used to report success).
- The set $\mathcal{O}$ (ranged over by $o, o', o'', E, E', \ldots$) of observers (tests) is defined like $\mathcal{P}$, where the grammar is extended with the production $P ::= \omega.P$.
- $\stackrel{\omega}{\longrightarrow}$ is the least predicate over $\mathcal{O}$ satisfying the inference rules in Table 3.

Omega $\quad \omega.E \stackrel{\omega}{\longrightarrow}$	Res $\quad \dfrac{E \stackrel{\omega}{\longrightarrow}}{(\nu y)E \stackrel{\omega}{\longrightarrow}}$
Par $\quad \dfrac{E_1 \stackrel{\omega}{\longrightarrow}}{E_1 \mid E_2 \stackrel{\omega}{\longrightarrow}}$	Cong $\quad \dfrac{E' \stackrel{\omega}{\longrightarrow} \quad E' \equiv E}{E \stackrel{\omega}{\longrightarrow}}$

Table 3
Predicate $\stackrel{\omega}{\longrightarrow}$.

Definition 3.2 [Maximal computations] Given $P \in \mathcal{P}$ and $o \in \mathcal{O}$, a maximal computation from $P \mid o$ is either an infinite sequence of the form

$$P \mid o = E_0 \longrightarrow E_1 \longrightarrow E_2 \longrightarrow \ldots$$

or a finite sequence of the form

$$P \mid o = E_0 \longrightarrow E_1 \longrightarrow \ldots \longrightarrow E_n \not\longrightarrow .$$

Definition 3.3 [May, must and fair relations [6]] Given $P \in \mathcal{P}$ and $o \in \mathcal{O}$, define:

- P *may* o if and only if *there is* a maximal computation (as in Def. 3.2) such that $E_i \stackrel{\omega}{\longrightarrow}$, for some $i \geq 0$;

- P *must* o if and only if *for every* maximal computation (as in Def. 3.2) there exists $i \geq 0$ such that $E_i \stackrel{\omega}{\longrightarrow}$;

- P *fair* o [6] if and only if *for every* maximal computation (as in Def. 3.2) and $\forall i \geq 0$, $\exists\, E_i'$ such that $E_i \stackrel{*}{\longrightarrow} E_i'$ and $E_i' \stackrel{\omega}{\longrightarrow}$.

4 Encoding linearity into persistence

First, we recall some notions about encodings. An encoding is a mapping from the terms of a calculus into the terms of another. In general a "good" encoding satisfies some additional requirements, but there is no agreement on a general notion of "good" encoding. Perhaps indeed there should not be a unique notion, but several, depending on the purpose. In this paper we shall study the existence of encodings $[\![\cdot]\!] : \mathrm{A}\pi \to \mathcal{P}$ from π into $\mathcal{P} \in \{\mathrm{PA}\pi, \mathrm{PIA}\pi, \mathrm{POA}\pi\}$ and focus on typical requirements such as compositionality w.r.t. certain operators, and the correctness w.r.t. a given semantics.

Compositionality and multi-hole contexts: We shall use notion of (multi-hole) process *contexts* [30] to describe compositionality. Recall that a $\mathcal{P}$ context C with k holes is a term with occurrences of k distinct *holes* $[\,]_1, \ldots, [\,]_k$ such that a $\mathcal{P}$ process must result from C if we replace all the occurrences of each $[\,]_i$ with a $\mathcal{P}$ process. The context C is *singularly-structured* if each hole occurs exactly once. For example, $[\,]_1 \mid x(y).([\,]_2 \mid [\,]_1)$ is an $\mathrm{A}\pi$ non singularly-structured context with two holes. Given $P_1, \ldots, P_k \in \mathcal{P}$ and a context C with k holes, $C[P_1, \ldots, P_k]$ is the process that results from replacing the occurrences of each $[\,]_i$ with P_i. The names of a context C with k holes, $n(C)$, are those of $C[Q_1, \ldots, Q_k]$ where each Q_i is 0. The free and bound names of a context are defined analogously. We can regard the input prefix $x(y)$, $\mid$ and ! as the operators of arity 1, 2 and 1 respectively in $\mathrm{A}\pi$ in the obvious sense.

Definition 4.1 [Compositionality w.r.t. an operator] Let op be an n-ary operator of $\mathrm{A}\pi$. An encoding $[\![\cdot]\!] : \mathrm{A}\pi \to \mathcal{P}$ is *compositional* w.r.t. op iff there is a $\mathcal{P}$ context C_{op} with n holes such that $[\![op(P_1, .., P_n)]\!] = C_{op}[[\![P_1]\!], .., [\![P_n]\!]]$.

[6] It may be possible to give other equivalent definition not based on maximal computations by using properties of the calculi under consideration such as: if $P \stackrel{\omega}{\longrightarrow}$ and $P \longrightarrow P'$ then $P' \stackrel{\omega}{\longrightarrow}$. For uniformity, however, we have used a well-known testing semantics definition based on the notion of *maximal* computations.

In the following, $C[\cdot]$ denotes contexts with one hole and $C[\cdot,\cdot]$ contexts with two holes. Furthermore, given an encoding $[\![\cdot]\!] : \mathrm{A}\pi \to \mathcal{P}$, we define ${C_{op}}^{[\![\cdot]\!]}$ as the context C such that $[\![op(P_1,\ldots,P_n)]\!] = C[[\![P_1]\!],\ldots,[\![P_n]\!]]$. We shall often omit the "$[\![\cdot]\!]$" in ${C_{op}}^{[\![\cdot]\!]}$ since it is easy to infer from the context.

Remark 4.2 [Homomorphism wrt parallel composition] An interesting case of compositionality is *homomorphism* w.r.t a given operator *op*: The operator is mapped into the same operator of the target language, i.e. $[\![op(P_1,..,P_n)]\!] = op([\![P_1]\!],\ldots,[\![P_n]\!])$. Homomorphism w.r.t parallelism, also called distribution-preserving [33,26,27], can arguably be considered as a reasonable requirement for an encoding. In particular, the works [33,26,27,23,9,16,17] support the distribution-preserving hypothesis by arguing that it corresponds to requiring that the degree of distribution of the processes is maintained by the translation, i.e. no coordinator is added. Some of these works are in the context of solving electoral problems and some others in more general scenarios [16,17]. Other works [22,28], however, argue that the requirement can be quite demanding as it rules out practical implementation of distributed systems. Some of our impossibility results will appeal to the distribution-preserving hypothesis.

Remark 4.3 Typically, the C_{op} mentioned in Definition 4.1 is a singularly-structured multi-hole context in encodings of operators such as input prefix, parallel composition and replication. Note that, if the encoding is homomorphic wrt *op*, then C_{op} is a singularly-structured multi-hole context.

Correctness wrt Testing: Concerning semantic correctness, we consider preservation of *sat* testing, where *sat* can be respectively *may*, *must* and *fair*. Given an encoding $e = [\![\cdot]\!] : \mathrm{A}\pi \to \mathcal{P}$, we assume that its lifted version e' from the set of observers of π to the ones of $\mathcal{P}$ is an encoding satisfying the following: $e'(o) = e(o)$ if o has no occurrences of ω.

Definition 4.4 [Soundness, completeness and *sat*-preservation] Let $[\![\cdot]\!] : \mathrm{A}\pi \to \mathcal{P}$. We say that $[\![\cdot]\!]$ is:

- *sound w.r.t. sat* iff $\forall\, P \in \mathrm{A}\pi,\ \forall\, o \in \mathcal{O},\ [\![P]\!]$ *sat* $[\![o]\!]$ implies P *sat* o;
- *complete w.r.t. sat* iff $\forall\, P \in \mathrm{A}\pi,\ \forall\, o \in \mathcal{O},\ P$ *sat* o implies $[\![P]\!]$ *sat* $[\![o]\!]$;
- *sat-preserving* iff $[\![\cdot]\!]$ is *sound* and *complete* w.r.t. *sat*.

4.1 Some encodings from asynchronous pi- calculus into its semi-persistent subsets

We consider the following encoding from $\mathrm{A}\pi$ to $\mathrm{PIA}\pi$ defined in [24].

Definition 4.5 The encoding $[\![\cdot]\!] : \mathrm{A}\pi \to \mathrm{PIA}\pi$ is a homomorphism for 0, parallel composition, restriction and replication, otherwise is defined

- $[\![\bar{x}z]\!] = \bar{x}z$, and
- $[\![x(y).P]\!] = (\nu t f)(\bar{t} \mid !x(y).(\nu l)(\bar{l} \mid !t.!l.([\![P]\!] \mid !\bar{f}) \mid !f.!l.\bar{x}y))$
  where $t, f, l \notin fn(P) \cup \{x, y\}$. (The lifted version is given adding $[\![\omega.P]\!] = \omega.[\![P]\!]$.)

This encoding enjoys a strong property: namely, for any P, $[\![P]\!] \approx P$, where $\approx$ denotes weak barbed congruence [30]. This implies, in the testing scenario, a

property stronger than *sat*-preservation.

Proposition 4.6 *Let* $[\![\cdot]\!] : A\pi \to PIA\pi$ *as in Definition 4.5.* $\forall\, P \in \mathrm{A}\pi$, $\forall\, o \in \mathcal{O} \subseteq \mathrm{PIA}\pi$ P *sat* o iff $[\![P]\!]$ *sat* o, where *sat* can be respectively *may* and *fair*.

To prove that the statement does not hold in the case of *must* semantics, consider $P = (a.0 \,|\, !\bar{a})$ and $o = a.\omega.0$: then P *must* o but $[\![P]\!]$ *m̸ust* o.

Extending the notion of barb to ω, Clearly $P \,|\, o \approx [\![P \,|\, o]\!]$ as $P \,|\, o \in \mathrm{A}\pi$, and by homomorphism w.r.t parallel composition, we obtain that $P \,|\, o \approx [\![P]\!] \,|\, [\![o]\!]$. This is enough to hold fair and may preserving.

In [24] the encoding in Definition 4.5 is used to get an encoding of $\mathrm{A}\pi$ into $\mathrm{POA}\pi$, by composing it with the following mapping from $\mathrm{PIA}\pi$ into $\mathrm{POA}\pi$.

Definition 4.7 The encoding $f = [\![\cdot]\!] : \mathrm{PIA}\pi \to \mathrm{POA}\pi$ is a homomorphism for 0, parallel composition, restriction, and replication, otherwise is defined as

- $[\![\bar{x}z]\!] = (\nu s)(!\bar{x}s \mid s(r).!\bar{r}z)$, and
- $[\![!x(y).P]\!] = !x(s).(\nu r)(!\bar{s}r \mid r(y).[\![P]\!])$

  where $s, r \notin fn(P) \cup \{x, z\}$. (The lifted version is given adding $[\![\omega.P]\!] = \omega.[\![P]\!]$.)

Let g be $[\![\cdot]\!] : \mathrm{A}\pi \to \mathrm{PIA}\pi$ in Definition 4.5. The encoding $h = [\![\cdot]\!] : \mathrm{A}\pi \to \mathrm{POA}\pi$ is the composite function $f \circ g$.

Because of this encoding maps a linear output into a replicated one with the same barb, the composite encoding $h = [\![\cdot]\!] : \mathrm{A}\pi \to \mathrm{POA}\pi$ in Definition 4.7 does not satisfy $[\![P]\!] \approx P$. It has a weaker property: namely, $P \approx Q$ iff $[\![P]\!] \approx \overset{\mathrm{POA}\pi}{[\cdot]} [\![Q]\!]$, where $[\![P]\!] \approx \overset{\mathrm{POA}\pi}{[\cdot]} [\![Q]\!]$ means that $\forall C[\cdot]$ context in $\mathrm{A}\pi$, $[\![C]\!][\![P]\!]]$ and $[\![C]\!][\![Q]\!]]$ (assuming $[\![[]\!]] = []$) are weak barbed bisimilar [30]. Similarly, the results for the composite encoding from $\mathrm{A}\pi$ into $\mathrm{POA}\pi$ in a testing scenario are weaker than these ones for the encoding from $\mathrm{A}\pi$ into $\mathrm{PIA}\pi$. Obviously, the following proposition would not hold if *sat* were *must*. Consider $P = !\bar{a}$ and $o = a.\omega.0$: then P *must* o but $[\![P]\!]$ *m̸ust* $[\![o]\!]$.

Proposition 4.8 Let $h = [\![\cdot]\!] : \mathrm{A}\pi \to \mathrm{POA}\pi$ as in Definition 4.7. $\forall P \in \mathrm{A}\pi, \forall o \in \mathcal{O}$, P *sat* o if and only if $[\![P]\!]$ *sat* $[\![o]\!]$, where *sat* can be respectively *may* and *fair*.

5 Uniform impossibility results for persistence

This section is the core of the paper and it focuses on general and uniform negative results for encodings of $\mathrm{A}\pi$ into $\mathrm{PIA}\pi, \mathrm{POA}\pi$ and $\mathrm{PA}\pi$, respectively. We identify some reasonable conditions which will guarantee that none of these encodings can be must-preserving. In particular, we show that there does not exist a must-preserving compositional encoding, homomorphic wrt replication, from π-calculus into any semi-persistent calculus. The proofs mainly rely on the following statement: if $[\![\cdot]\!]$ is an encoding from $\mathrm{A}\pi$ into $\mathcal{P}$ satisfying (1) compositionality w.r.t. input prefix, (2) *must*-preservation and (3) $[\![\omega.0]\!] \xrightarrow{\omega}$ then $\forall x, y \in \mathcal{N}$, any hole is prefixed in $C^{[\![\cdot]\!]}_{x(y)}$.

We believe that the hypothesis $[\![\omega.0]\!] \xrightarrow{\omega}$ is reasonable for an encoding. It can follow from the existence of a divergent process in the range of the encoding which

is necessary if the encoding preserves divergence—recall that P *diverges*, $P\uparrow$, if there is an infinite sequence of reductions from P. However, it can be a divergence-independent property of the encoding, obtained in a purely syntactic way whenever the lifted version of the encoding is defined adding $[\![\omega.P]\!] = \omega.[\![P]\!]$.

Theorem 5.1 Let $[\![\cdot]\!] : \mathrm{A}\pi \to \mathcal{P}$, with $\mathcal{P} \in \{\mathrm{PIA}\pi, \mathrm{POA}\pi, \mathrm{PA}\pi\}$, be an encoding satisfying:

1. compositionality w.r.t. input prefix, parallelism and replication,
2. $[\![\omega.0]\!] \xrightarrow{\omega}$,
3. $\exists x, y, z :\ n(C_!^{[\![\cdot]\!]}) \cap n(C_{x(y)}^{[\![\cdot]\!]}) = n(C_!^{[\![\cdot]\!]}) \cap n([\![\bar{x}z]\!]) = n(C_!^{[\![\cdot]\!]}) \cap n(C_|^{[\![\cdot]\!]}) = \emptyset$,
4. $C_!^{[\![\cdot]\!]}$ is a singularly-structured context.

Then $[\![\cdot]\!]$ is not *must*-preserving.

Proof. (Sketch of:) Suppose that $[\![\cdot]\!]$ in $C_!^{[\![\cdot]\!]}$ is not in the scope of a replication. Then it is possible to prove that the hole is prefixed in $C_!^{[\![\cdot]\!]}$. Now it suffices to consider that $x(y).0$ *must* $!\omega.0$ but $C_{x(y)}[[\![0]\!]]$ $\not$*must* $C_![[\![\omega.0]\!]]$, since every hole is prefixed in $C_{x(y)}^{[\![\cdot]\!]}$, the hole is prefixed in $C_!^{[\![\cdot]\!]}$ and $C_{x(y)}[[\![0]\!]] \mid C_![[\![\omega.0]\!]] \longrightarrow$ by (3).

Now suppose that $[\![\cdot]\!]$ in $C_!^{[\![\cdot]\!]}$ is in the scope of a replication. Then it is possible to prove that $\forall x', z' \in \mathcal{N}$, either $C_![C_{x(y)}[[\![\omega.0]\!]]] \mid [\![\bar{x}z \mid \bar{x}'z']\!]$ or $C_{x(y)}[[\![\omega.0]\!]] \mid C_![[\![\bar{x}z \mid \bar{x}'z']\!]]$ has at least one infinite computation such that $[\![\omega.0]\!]$ does not interact or participate in the computation. Now it suffices to consider both $P \mid o$ (with $[\![P]\!] \mid [\![o]\!]$) and $P' \mid o'$ (with $[\![P']\!] \mid [\![o']\!]$), where $P = !x(y).x'(y').\omega.0$, $o = \bar{x}z \mid \bar{x}'z'$ $(x \neq x')$, $P' = x(y).x'(y').\omega.0$ and $o' = !(\bar{x}z \mid \bar{x}'z')$, obtaining that $[\![\cdot]\!]$ cannot be *must*-preserving.□

Let us discuss the premises in the above theorem. Compositionality is in general a reasonable condition for an encoding. As argued above, the second condition is validated if the encoding is to preserve divergence. The third condition is validated if in the encoding of each operator *op* the context where the encodings of the operands are placed, i.e. C_{op}, uses unique names only. Replication represents an infinite parallel composition, so it is arguably reasonable to require homomorphism for replication since homomorphism for the parallel operator is arguably a reasonable requirement—see Remark 4.2. Regarding (4), we already pointed out in Remark 4.3 that in compositional encodings the contexts C_{op} are typically singularly-structured [7] .

We conclude this section with a theorem stating a general and uniform impossibility result for the existence of encodings from $\mathrm{A}\pi$ into any (semi-)persistent calculus. The statement results as an immediate consequence of Theorem 5.1 in the case of homomorphism w.r.t replication, as it implies $n(C_!^{[\![\cdot]\!]}) = \emptyset$.

Theorem 5.2 Let $[\![\cdot]\!] : \mathrm{A}\pi \to \mathcal{P}$, with $\mathcal{P} \in \{\mathrm{PIA}\pi, \mathrm{POA}\pi, \mathrm{PA}\pi\}$, be an encoding satisfying:

1. compositionality w.r.t. input prefix and parallelism,

[7] Notice that the situation pointed out to us a previous review where $[\![!P]\!] = [\![P]\!] \mid ![\![P]\!]$, i.e., $C_! = [.] \mid ![.]$ is not singularly-structured, can be rewritten via $\equiv$ as $[\![!P]\!] = ![\![P]\!]$ and in this case the corresponding $C_! = ![.]$ is singularly-structured.

2. homomorphism w.r.t bang operator,
3. $[\![\omega.0]\!] \xrightarrow{\omega}$.

Then $[\![\cdot]\!]$ is not *must*-preserving.

6 Specialized impossibility results for persistence

In the previous section we gave a uniform impossibility result for the existence of encodings of Aπ into the (semi-)persistent calculi. In this section, we give further impossibility results, under different hypotheses, taking into account particular features of some of the (semi-)persistent calculi, namely PAπ and PIAπ [8] .

For technical reasons we introduce a particular kind of contexts in $\mathcal{P}$ that differ from those we have introduced in Section 4, in that brackets do not disappear once we "fill the holes" with process terms. Additionally, we require that different occurrences of braces are to be filled with the same process.

Definition 6.1 [Focusing contexts] A *focusing context* $C\{\ \}$ for $\mathcal{P}$ is generated by the following grammar:

$$C\{\ \} := \ \{\ \}\sigma \ \Big|\ 0 \ \Big|\ out \ \Big|\ in.C\{\ \} \ \Big|\ (\nu x)C\{\ \} \ \Big|\ C\{\ \} \mid C\{\ \} \ \Big|\ !C\{\ \}$$

where σ is a (name) substitution, and *in* and *out* are resp. input and output, according to $\mathcal{P}$ syntax. (e.g. $in = !x(y)$ and $out = \bar{x}z$ when $\mathcal{P} = \text{PIA}\pi$)

Notation 6.1 *Given a focusing context $C\{\}$ and $P \in \mathcal{P}$, $C\{P\}$ is the term obtained by replacing each occurrence $\{\ \}\sigma$ in $C\{\ \}$ by $\{P\}\sigma$. We denote by $\mathcal{L}(P)$ (ranged over by $B, B', ..$) the set $\{C\{P\} \mid P \in \mathcal{P}, C\{\ \}$ is a focusing context$\}$.*

An occurrence of $\{P\}\sigma$ is prefixed in $B \in \mathcal{L}(P)$ if it is in the scope of an input prefix. We write Pref(B) when every occurrence of $\{P\}\sigma$ is prefixed in B.

The structural equivalence and the reduction semantics for the language $\mathcal{L}(P)$ are both defined on the basis of the ones for $\mathcal{P}$, the only difference being that terms are in $\mathcal{L}(P)$ instead than in $\mathcal{P}$ and that unguarded braces (i.e. terms out of the scope of an input prefix like $\{P\}\sigma$) are assumed as deadlocked terms. This is not a concern, because for the proof of our main results, for every σ each occurrence of $\{P\}\sigma$ is prefixed, i.e. in the scope of an input prefix.

It is possible to prove that $\mathcal{L}(P)$ is closed under substitution and, as a consequence, under reduction. Denoting by *Unbrace*(B) the $\mathcal{P}$ process obtained by removing all the braces from B and by applying the substitutions, it is also possible to prove that: (i)$B \in \mathcal{L}(P)$, then (i) $B \longrightarrow B'$ implies $B' \in \mathcal{L}(P)$ and *Unbrace*$(B) \longrightarrow$ *Unbrace*(B'), and (ii) *Pref*(B) and *Unbrace*$(B) \longrightarrow R$ implies that $\exists B' \in \mathcal{L}(P)$ such that $B \longrightarrow B'$ and $R \equiv$ *Unbrace*(B').

Focusing contexts are extended for the testing machinery, adding rule $\{\omega.E\}\iota \xrightarrow{\omega}$ in Table 3. Notice that, since every σ is defined over $\mathcal{N}$ and $\omega \notin \mathcal{N}$, then $\forall E \in \mathcal{P}$ and $B \in \mathcal{L}(P)$, (i) $\{\omega.E\}\sigma \xrightarrow{\omega}$; (ii) $B \xrightarrow{\omega}$ implies $B\sigma \xrightarrow{\omega}$; (iii) $B \xrightarrow{\omega}$ if and only if *Unbrace*$(B) \xrightarrow{\omega}$, where $B\sigma$ represents the result of the application of σ to B (assuming to use α-equivalence to avoid collision of names).

[8] We also stated this kind of specialized result for POAπ but for reasons of space and its restricted nature it has been moved to the full paper appendix

Persistent Pi-Calculus: To prove our main results, we define a function over $\mathcal{L}(P)$, $min(B)$ (Table 4), and a predicate, Pr (Table 5).

$min(B) = +\infty$ if $B \in \mathcal{P}$;	$min((\nu x)B) = min(B)$;
$min(\{P\}) = 0$;	$min(B \mid B') = \mathbf{min}\{min(B), min(B')\}$;
$min(x(y).B) = 1 + min(B)$;	$min(!B) = min(B)$.

Table 4
Function min.

$$\text{Red}\ \frac{min(!x(y).B) \geq 2}{Pr(!\bar{x}z \mid !x(y).B)} \qquad \text{Res}\ \frac{Pr(B)}{Pr((\nu y)B)}$$

$$\text{Par}\ \frac{Pr(B_1)}{Pr(B_1 \mid B_2)} \qquad \text{Cong}\ \frac{Pr(B')\ ,\ B' \equiv B}{Pr(B)}$$

Table 5
Predicate Pr.

We can prove that Pr is closed under reduction and it implies $Pref$. As a consequence, for every $B \in \mathcal{L}(P)$ such that $Pr(B)$, it is possible to build a non-empty maximal computation from B where any term of the computation verifies the predicate Pr. We can now state a rather strong negative result for PAπ.

Theorem 6.2 Let $[\![\cdot]\!]$ be an encoding from Aπ into PAπ that satisfies:

1. compositionality w.r.t. input prefix,
2. $[\![\omega.0]\!] \xrightarrow{\omega}$.

Then $[\![\cdot]\!]$ is not *must*-preserving.

Proof. By contradiction, it suffices to suppose $[\![\cdot]\!]$ being *must*-preserving, consider $P = \bar{x}z|\bar{x}z$ and $o = x(y).x(y).\omega.0.$ and observe that $Pr([\![\bar{x}z|\bar{x}z]\!]|C_{x(y)}[C_{x(y)}\{[\![\omega.0]\!]\}])$ holds. Hence, it is possible to prove that there is a non-empty maximal computation from $[\![\bar{x}z \mid \bar{x}z]\!] \,|[\![x(y).x(y).\omega.0]\!]$ where any term of the computation verifies the predicate Pr, i.e. every term does not perform ω (since every occurrence of $[\![\omega.0]\!]$ is prefixed). □

The above theorem resembles the impossibility result in [24] about the existence of an encoding from Aπ into PAπ wrt weak bisimulation (and output equivalence). However, the hypothesis of the result in [24] is different. Namely, it is restricted to encodings homomorphic wrt parallel composition.

Persistent-Input and Persistent-Output Pi-Calculus: Regarding both the semi-persistent calculi, a Pr-like predicate does not preserve $Pref$ (it suffices to consider $B_1 = \bar{b} \mid \bar{c} \mid !b.!c.\{P\}\sigma$, where $P \in$ PIAπ, and $B_2 = !\bar{b} \mid !\bar{c} \mid b.c.\{P\}\sigma$, where $P \in$ POAπ). In the case of PIAπ, an ad-hoc predicate, Pr_{in}, is defined. The predicate has been defined in such a way to select those processes $B \in \mathcal{L}(P)$ such that - every $\{P\}\sigma$ occurrence is in the scope of an input prefix $x(y)$, for some

$x \in fn(B)$ and $y \in \mathcal{N}$, - there exists an input component $!x(y).B$ (prefixing $\{P\}\sigma$) such that $min(!x(y).B) \geq 2$, and - every parallel component $!x_i(y).B$ is such that $min(!x_i(y).B) \geq 1$ if $x_i = x$ and $min(!x_i(y).B) \geq 2$ if $x_i \neq x$. As Pr, Pr_{in} preserves *Pref* under reduction as well as the other results for Pr. In particular, whenever $\exists x \in fn(B)$ such that $Pr_{in}(B, x)$, it is possible to build a maximal computation from B where any term of the computation verifies the predicate Pr_{in}. Hence, it leads us to the negative result below.

Theorem 6.3 Let $[\![\cdot]\!]$ be an encoding from Aπ into PIAπ that satisfies:

1. compositionality w.r.t. input prefix,
2. $[\![\omega.0]\!] \xrightarrow{\omega}$,
3. if $fn(P) \cap bn(x(y)) = \emptyset$ then $fn([\![P]\!]) \cap bn(C^{[\![\cdot]\!]}_{x(y)}) = \emptyset$,
4. $[\![x(y).P]\!] \equiv (\nu\vec{x})((!u(v).C[[\![P]\!]]) \mid T)$ for some $\vec{x}, C, T$ with $u \notin \vec{x}$.

Then $[\![\cdot]\!]$ is not *must*-preserving.

Proof. It is possible to prove that $\exists h \in fn(C^{[\![\cdot]\!]}_{x(y)})$: $Pr_{in}(C_{x(y)}[C_{x(y)}\{[\![\omega.0]\!]\}], h)$. Now, it suffices to assume, by contradiction, that $[\![\cdot]\!]$ is *must*-preserving and proving that $Pr_{in}([\![\bar{x}z \mid \bar{x}z]\!] \mid C_{x(y)}[C_{x(y)}\{[\![\omega.0]\!]\}], h)$ holds. Hence, it is possible to prove that there is a non-empty maximal computation from $[\![\bar{x}z \mid \bar{x}z]\!] \mid [\![x(y).x(y).\omega.0]\!]$ where any term of the computation verifies the predicate Pr_{in}, i.e. every term does not perform ω (since every occurrence of $[\![\omega.0]\!]$ is prefixed). □

Notice that the encoding in Definition 4.5 satisfies every condition of the following theorem and, more important, that Pr_{in} does not rely on any divergence assumption, differently from Pr. We have already argued for the first two conditions as being reasonable. Intuitively, the third condition expresses that the encoding preserves *non-binding* wrt input prefix: If in a source term $x(y).P$ none of the free names of P is bound by the input prefix, then the free names of $[\![P]\!]$ must not be bound either (by a binder in the context where $[\![P]\!]$ is placed) in the encoding of $[\![x(y).P]\!]$. Finally, the fourth condition basically expresses that Aπ inputs should be mapped into PIAπ inputs possibly allowing some other material around it. This is validated, e.g., by encodings that preserve input/output polarities—i.e., Aπ inputs/outputs must be mapped into PIAπ input/outputs [9].

7 Related Work and Concluding Remarks

Most of the related work was discussed in the introduction. In a different context, in [22] it is shown that the separate choice encoding of the π-calculus into the asynchronous π-calculus is faithful with respect to weak bisimulation, while in [8] the authors prove that no must-preserving encoding of the (choiceless) synchronous pi-calculus into the asynchronous one exists. Hence must semantics is a good candidate to study the expressiveness of persistence when divergence is taken into account. Nevertheless, differently from [8], this work does not consider any

[9] E.g., the encoding in Definition 4.5 satisfies all conditions of Theorem 6.3.

synchronous language, the must semantics is studied in a uniform and purely asynchronous framework. As previously mentioned the study of persistence in [24] is incomplete as ignores the crucial issue of divergence. In this paper, we used the divergence-sensitive framework of testing semantics and adapted and exploited the techniques of [8] to give a more complete account of the expressiveness of persistence in asynchronous calculi. In particular, as discussed in the introduction, this work supports informal expressiveness loss claims in persistent asynchronous languages [3,14,11].

References

[1] J. Alves-Foss. An Efficient Secure Authenticated Group Key Exchange Algorithm for Large and Dynamic Groups. In *Proceedings of the 23rd National Information Systems Security Conference*, 2000.

[2] R. Amadio and D. Lugiez and V. Vanackere. On the Symbolic Reduction of Processes with Cryptographic Functions. *TCS: Theoretical Computer Science* 290, 2003.

[3] E. Best, F. de Boer, and C. Palamidessi. Partial order and sos semantics for linear constraint programs. *In Proc. of Coordination'97*, volume 1282 of LNCS, 1997.

[4] B. Blanchet. From linear to classical logic by abstract interpretation. *Information Processing Letters* 95(5), 2005.

[5] M. Boreale and M. Buscemi. A Framework for the Analysis of Security Protocols, *Lecture Notes in Computer Science* 2421, 2002.

[6] E. Brinksma, A. Rensink, W. Vogler. Fair Testing, *Proc. of CONCUR'95, LNCS* 962, pp. 313-327, 1995.

[7] D. Cacciagrano, F. Corradini. On Synchronous and Asynchronous Commu- nication Paradigms, *Proc. of ICTCS '01, LNCS* 2202, pp. 256-268, 2001.

[8] D. Cacciagrano, F. Corradini, C. Palamidessi. Separation of Synchronous and Asynchronous Communication Via Testing. *Proc. of EXPRESS'05*. Electr. Notes Theor. Comput. Sci. 154(3): 95-108, 2006. An extended version will appear in *Theoretical Computer Science*.

[9] M. Carbone, S. Maffeis. On the Expressive Power of Polyadic Synchronisation in pi-calculus. *Nord. J. Comput.* 10(2): 70-98, 2003.

[10] I. Castellani, M. Hennessy. Testing Theories for Asynchronous Languages, *Proc. of FSTTCS '98, LNCS* 1530, pp. 90-101, 1998.

[11] F. Crazzolara. Language, Semantics, and Methods for Security Protocols. *PhD Dissertation*, University of Aarhus, Denmark, 2003.

[12] F. Crazzolara and G. Winskel. Events in security protocols, *Proceedings of the 8th ACM Conference on Computer and Communications Security*, ACM Press, 2001.

[13] R. De Nicola, M. Hennessy. Testing Equivalence for Processes, *Theoretical Computer Science* 34, pp. 83-133, 1984.

[14] F. Fages, P. Ruet, and S. Soliman. Linear concurrent constraint programming: operational and phase semantics. *Information and Computation*, 2001.

[15] M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. *Proc. CSFW-14. IEEE*, 2001.

[16] D. Gorla: On the Relative Expressive Power of Asynchronous Communication Primitives. *FoSSaCS 2006*, 47-62, 2006.

[17] D. Gorla: Synchrony vs Asynchrony in Communication Primitives *Proc. of EXPRESS'06*, 47-62, 2006.

[18] S. Maffeis and I. Phillips. On the computational strength of pure ambient calculi. *Proc. of EXPRESS '03*, 2003.

[19] R. Milner. *Communication and Concurrency*, Prentice-Hall International, 1989.

[20] R. Milner, J. Parrow, D. Walker. A Calculus of Mobile Processes, Part I and II, *Information and Computation* 100, pp. 1-78, 1992.

[21] M. Merro, D. Sangiorgi. On asynchrony in name-passing calculi, *Proc. of ICALP '98, LNCS* 1443, 1998.

[22] U. Nestmann. What is a 'Good' Encoding of Guarded Choice?, *Information and Computation* 156, pp. 287-319, 2000.

[23] C. Palamidessi. Comparing the Expressive Power of the Synchronous and Asynchronous π-calculus, *Mathematical Structures in Computer Science* 13(5), pp. 685-719, 2003. A preliminary version appeared in the proceedings of POPL '97.

[24] C. Palamidessi, V. Saraswat, F. Valencia and B. Victor. On the Expressiveness of Linearity vs Persistence in the Asynchronous Pi Calculus. *LICS 2006:59-68*, 2006.

[25] L. C. Paulson. Mechanized proofs for a recursive authentication protocol. In *10th Computer Security Foundations Workshop*, 1997.

[26] I. Phillips and M. Vigliotti Electoral Systems in Ambient Calculi. *FoSSaCS'04*. 2004.

[27] I. Phillips, M. Vigliotti. Leader Election in Rings of Ambient Processes. *Electr. Notes Theor. Comput. Sci.* 128(2): 185-199, 2005.

[28] K.V.S. Prasad. Broadcast Calculus Interpreted in CCS up to Bisimulation. *In Proceedings of Express'01*, volume 52 of Electronic Notes in Theoretical Computer Science, pages 83-100. Elsevier, 2002.

[29] D. Sangiorgi. The name discipline of uniform receptiveness. *Theoretical Computer Science*, 221(12):457493, 1999.

[30] D. Sangiorgi and D. Walker. *The π-calculus: A Theory of Mobile Processes.* Cambridge University Press, 2001.

[31] V. Saraswat and P. Lincoln. Higher-order linear concurrent constraint programming. *Technical report, Xerox PARC*, 1992.

[32] V. Saraswat. Concurrent Constraint Programming. The MIT Press, 1993.

[33] M. Vigliotti, I. Phillips, C. Palamidessi. Separation Results Via Leader Election Problems. *FMCO 2005*, 172-194, 2005.

Models of Computation: Automata and Processes

Invited talk

Jos Baeten[1,2]

Formal Methods Group
Division of Computer Science
Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, NL-5600 MB Eindhoven
The Netherlands

Abstract

The computational models of automata theory and concurrency theory have a lot in common: there is an underlying model with states and transitions, a grammar corresponds to a recursive specification and the algebra of regular expressions is a process algebra. More can be done to exploit this commonality, results, methods and techniques and questions can be transfered from one domain to the other, and standardization in the concurrency domain can be furthered. To give an example, we know that every computable process is equal to a regular process communicating with a bidirectional tape (a Turing machine) modulo weak or branching bisimulation. We establish that every context-free process is equal to a regular process communicating with a stack modulo weak or branching bisimulation.

[1] Joint work with Bas Luttik and Clemens Grabmayer.

[2] Email: josb@win.tue.nl

This is a preliminary version. The final version will be electronically published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.com/locate/entcs

Alternating-time temporal logic: expressivity, complexity,... and variants

Invited talk

François Laroussinie[1]

Laboratoire Spécification et Vérification
CNRS UMR 8643
École Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 CACHAN Cedex - France

Abstract

In this talk, we will consider the Alternating-time Temporal Logic (ATL). We will present several results about its expressivity and complexity depending the kind of multi-agent model used to interpret ATL formulae (Concurrent Game Structures, Alternating Transition Systems, Implicit Concurrent Game Structures,...). We will also present several real-time extensions of ATL and its models. In particular we will discuss the way of combining concurrency and time.

[1] Email: fl@lsv.ens-cachan.fr

This is a preliminary version. The final version will be electronically published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.com/locate/entcs

Leader Election and Expressiveness

Invited talk

Iain Phillips[1,2]

Theory of Computational Systems Section
Department of Computing
Imperial College, London
United Kingdom

Abstract

Choosing a leader process in a network of processes is a classical problem of distributed computing. The work of Boug and Palamidessi has shown that we can regard the ability to perform leader election in a symmetric and distributed fashion as a measure of the expressive power of process languages. We discuss the results which have been obtained following this line of research, and compare them with alternative approaches. In particular, we consider ambient calculi (languages which allow mobile code to be transported in containers called "ambients"). We also describe current investigation (with MohammadReza Mousavi) into the properties of general process languages which determine whether leader election is possible.

[1] Joint work with Maria Grazia Vigliotti.

[2] Email: iccp@doc.ic.ac.uk

This is a preliminary version. The final version will be electronically published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.com/locate/entcs

When is an encoding good? Full abstraction and other criteria.

Discussion introduced and moderated by

Daniele Gorla and Uwe Nestmann

Abstract

One of the hottest topic for the EXPRESS audience is the definition of the criteria that make an encoding 'good': this is a crucial issue to accept an encoding or to prove separation results. Here are some specific problems we will discuss. What is the the relevance of operational correspondence in proving separation results? What kind of renamings should an encoding preserve? Is it mandatory to map parallel composition homomorphically?

This is a preliminary version. The final version will be electronically published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.com/locate/entcs