On the relationship between π-calculus and finite place/transition Petri nets[*]

Roland Meyer[1] and Roberto Gorrieri[2]

[1] LIAFA, Paris Diderot University & CNRS
e-mail: `roland.meyer@liafa.jussieu.fr`

[2] Department of Computing Science, University of Bologna
e-mail: `gorrieri@cs.unibo.it`

Abstract. We clarify the relationship between π-calculus and finite p/t Petri nets. The first insight is that the concurrency view to processes taken in [Eng96,AM02,BG09] and the structural view in [Mey09] are orthogonal. This allows us to define a new concurrency p/t net semantics that can be combined with the structural semantics in [Mey09]. The result is a more expressive mixed semantics, which translates precisely the so-called mixed-bounded processes into finite p/t nets. Technically, the translation relies on typing of restricted names. As second main result we show that mixed-bounded processes form the borderline to finite p/t nets. For processes just beyond this class reachability becomes undecidable and so no faithful translation into finite p/t nets exists.

1 Introduction

There has been considerable recent interest in verification techniques for mobile systems that are based on automata-theoretic and in particular Petri net translations [AM02,FGMP03,KKN06,DKK08,MKS09,BG09,Mey09]. Most verification approaches and tool implementations have been developed for finite place/transition (p/t) Petri nets. This raises the question for a characterisation of the processes that can be verified with the help of finite p/t Petri net semantics. Despite the efforts in the semantics community dating back to [BG95,MP95,Eng96,Pis99] the problem is still open and we tackle it in this paper. The contribution is the precise borderline that separates π-calculus processes from finite p/t Petri nets in terms of computational expressiveness.

In structurally stationary processes [Mey09] restricted names generate a finite number of connection graphs at runtime. Although the constraint is semantical and thus undecidable, large syntactic subclasses of structurally stationary processes exist, e.g. finite control [Dam96] and restriction-free processes [AM02]. The so-called structural semantics represents structurally stationary processes as finite p/t nets and is sound wrt. the reaction semantics [Mey09]. Therefore,

[*] Appeared in Proc. of CONCUR, volume 5710 of LNCS, pages 463–480. Springer-Verlag, 2009. `http://dx.doi.org/10.1007/978-3-642-04081-8_31`.

we take structurally stationary processes as starting point in our quest for the borderline to finite p/t nets.

Mobile systems are concurrent systems where the connections between components evolve over time. While the structural semantics focuses on these connections, concurrency semantics highlight the interactions between components. In [BG95,BG09], we defined a concurrency semantics that reflects the intended causality semantics for π-calculus. It is finite for the semantic class of restriction-bounded processes, which generate a finite number of restricted names. In this work, we present the largest class of processes with a finite p/t net semantics that is sound wrt. the reaction behaviour. We find it as follows.

We observe that the structural view to processes is orthogonal to the concurrency view. The main result is that an appropriately defined concurrency semantics can be combined with the structural semantics to a more expressive mixed translation. Intuitively, the new translation mirrors the interactions between groups of components. Technically, the combination is achieved by typing restricted names, and the type determines the semantics that handles a name.

We prove the mixed semantics to be finite precisely for mixed-bounded processes, which combine the previous finiteness characterisations, i.e., they form finitely many connection graphs with names of type one and generate finitely many restricted names of type two. Again, mixed boundedness is an undecidable semantic condition, but it inherits all syntactic subclasses of structurally stationary and restriction-bounded processes.

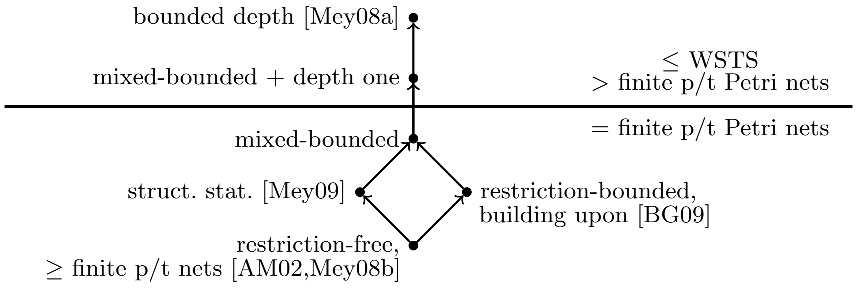
bounded depth [Mey08a]
mixed-bounded + depth one
≤ WSTS
> finite p/t Petri nets
= finite p/t Petri nets
mixed-bounded
struct. stat. [Mey09]
restriction-bounded, building upon [BG09]
restriction-free, ≥ finite p/t nets [AM02,Mey08b]

Fig. 1. Hierarchy of process classes and relation to finite p/t Petri nets ($\rightarrow := \subseteq$).

We then show that mixed-bounded processes form the *borderline* between π-calculus and finite p/t nets, since in a minimal extension of the class reachability becomes undecidable. Unfortunately, this extension lies within the processes of bounded depth, which are known to have well-structured transition systems (WSTS) and so, e.g. a decidable termination problem [Mey08a]. Hence, our results dash hope of a finite p/t net translation for this interesting class.

Note that every finite p/t net can be represented by a restriction-free process of linear size with contraction-isomorphic transition system [AM02,Mey08b].

Hence, all process classes we consider are at least as expressive as finite p/t nets. Figure 1 illustrates the elaborated relationships. We summarise our contribution.

– We define a new concurrency semantics for the π-calculus, which satisfies three indispensable quality criteria. It yields a bisimilar transition system, translates processes with restricted names, and enjoys an intuitive finiteness characterisation. The technical tool that facilitates the definition is a new *name-aware transition system*, which manages the use of restricted names.
– We combine the concurrency semantics with the structural semantics and prove the resulting translation finite precisely for so-called *mixed-bounded processes*. The idea to combine the semantics is to type restricted names. Technically, the definition also relies on a new normal form for processes under structural congruence.
– We prove that mixed-bounded processes form the borderline to finite p/t nets. If we relax the requirement slightly, reachability becomes undecidable.

Related Work Although several automata-theoretic semantics for the π-calculus have been proposed [Eng96,MP95,Pis99,AM02,BG95,BG09,KKN06,DKK08], we found them all defective in the sense that they do not satisfy the criteria we require for a semantics to be usable for verification. We discuss their problems.

Engelfriet [Eng96] translates processes with replication into p/t nets that are bisimilar to the reaction semantics. Since the Petri net representation is infinite as soon as the replication operator is used, the requirement for finiteness is not satisfied. In subsequent papers [EG99,EG04], Engelfriet and Gelsema show that the discriminating power of their semantics corresponds to extended and decidable versions of structural congruence. In the proofs, they exploit normal forms for processes similar to the restricted form we present in Section 2.

Montanari and Pistore propose history dependent automata (HDA) as semantic domain, finite automata where states are labelled by sets of names that represent the restrictions in use [MP95,Pis99]. The ground and early labelled transition semantics are translated into HDA in a way that bisimilarity on the automata coincides with the corresponding bisimilarity on processes. The translations yield finite HDA only for finitary processes, the subclass of structurally stationary processes obtained by bounding the degree of concurrency [Mey09].

Amadio and Meyssonnier [AM02] translate restriction-free processes into bisimilar (reaction semantics) and finite p/t nets. To deal with a class of processes that contain restrictions, a second translation identifies restricted names as unsed and replaces them by generic free names. Since the number of processes to be modified by replacement is not bounded, these authors rely on Petri nets with transfer as semantic domain. Having an undecidable reachability problem, transfer nets are strictly more expressive than finite p/t nets [DFS98].

Our work [BG95] translates the early labelled transition relation of restriction-bounded processes into finite p/t Petri nets, but fails to prove bisimilarity. Based on that translation, [BG09] studies non-interleaving and causal semantics for the π-calculus and provides decidability results for model checking.

Koutny et. al. [DKK08] achieve a bisimilar translation of the indexed labelled transition system into finite but high-level Petri nets, thus relying on a Turing complete formalism where automatic analyses are necessarily incomplete. The main contribution is compositionality, for every π-calculus operator there is a corresponding net operator and in many cases the size of the net is linear in the size of the process. In [KKN06], the translation is extended by an unfolding-based model checker. To avoid the undecidability the verification approach is restricted to recursion-free processes, a class of limited practical applicability.

2 Preliminaries

We recall the basics on π-calculus, p/t Petri nets, and the structural semantics.

π-calculus We use a π-calculus with parameterised recursion as proposed in [SW01]. Let the set $\mathcal{N} := \{a, b, x, y, \ldots\}$ of *names* contain the channels, which are also the possible messages, that occur in communications. During a process execution the *prefixes* π are successively removed from the process to communicate with other processes or to perform silent actions. The *output action* $\pi = \overline{a}\langle b\rangle$ sends the name b along channel a. The *input action* $\pi = a(x)$ receives a name that replaces x on a. Prefix $\pi = \tau$ performs a *silent action.*

To denote recursive processes, we use *process identifiers* $K, L, \ldots$ A process identifier is defined by an equation $K(\tilde{x}) := P$, where $\tilde{x}$ is a short-hand notation for $x_1, \ldots, x_k$. When the identifier is *called*, $K\lfloor\tilde{a}\rfloor$, it is replaced by process P with the names $\tilde{x}$ changed to $\tilde{a}$. More precisely, a *substitution* $\sigma = \{\tilde{a}/\tilde{x}\}$ is a function that maps the names in $\tilde{x}$ to $\tilde{a}$, and is the identity for all the names not in $\tilde{x}$. The *application of a substitution* is denoted by $P\sigma$ and defined in the standard way [SW01]. A π-calculus process is either a call to an identifier, $K\lfloor\tilde{a}\rfloor$, a *choice process* deciding between prefixes, $M + N$, a *parallel composition* of processes, $P_1 \mid P_2$, or the *restriction* of a name in a process, $\nu a.P$:

$$M ::= \mathbf{0} \;\mid\; \pi.P \;\mid\; M + N \qquad\qquad P ::= M \;\mid\; K\lfloor\tilde{a}\rfloor \;\mid\; P_1 \mid P_2 \;\mid\; \nu a.P.$$

Writing π for $\pi.\mathbf{0}$ we omit pending $\mathbf{0}$ processes. By $M^{=\mathbf{0}}$ we denote choice compositions of stop processes $\mathbf{0} + \ldots + \mathbf{0}$. To indicate a choice composition contains at least one term $\pi.P$ we denote it by $M^{\neq\mathbf{0}}$. Processes $M^{\neq\mathbf{0}}$ and $K\lfloor\tilde{a}\rfloor$ are called *sequential*, and they are the basic processes that produce reactions, either alone or by synchronisation of two of them, in the reaction relation defined below.

A restriction νa that is not covered by a prefix π is *active* and the *set of active restrictions* in a process is $arn(P)$. For example, $arn(\nu a.\overline{a}\langle b\rangle.\nu c.\overline{a}\langle c\rangle) = \{a\}$. The input action $a(b)$ and the restriction $\nu c.P$ *bind* the names b and c, respectively. The *set of bound names* in a process P is $(arn(P) \subseteq)\ bn(P)$. A name which is not bound is *free* and the *set of free names* in P is $fn(P)$. We permit α-conversion of bound names. Therefore, wlog. we assume that a name is bound at most once in a process and that $bn(P) \cap fn(P) = \emptyset$. Moreover, if a substitution $\sigma = \{\tilde{a}/\tilde{x}\}$ is applied to a process P, we assume $bn(P) \cap (\tilde{a} \cup \tilde{x}) = \emptyset$.

We use the *structural congruence* relation in the definition of the behaviour of a process term. It is the smallest congruence where α-conversion of bound names is allowed, $+$ and $\mid$ are commutative and associative with $\mathbf{0}$ as the neutral element, and the following laws for restriction hold:

$$\nu x.\mathbf{0} \equiv \mathbf{0} \qquad \nu x.\nu y.P \equiv \nu y.\nu x.P \qquad \nu x.(P \mid Q) \equiv P \mid (\nu x.Q), \text{ if } x \notin \mathit{fn}(P).$$

The last rule is called *scope extrusion.* The behaviour of π-calculus processes is then determined by the *reaction relation* $\rightarrow\ \subseteq \mathcal{P} \times \mathcal{P}$ defined by the rules in Table 1. By $\mathit{Reach}(P)$ we denote the *set of all processes reachable from* P by

$$\text{(Tau)} \quad \tau.P + M \rightarrow P \qquad \text{(React)} \quad x(y).P + M \mid \overline{x}\langle z\rangle.Q + N \rightarrow P\{z/y\} \mid Q$$

$$\text{(Const)} \quad K\lfloor\tilde{a}\rfloor \rightarrow P\{\tilde{a}/\tilde{x}\}, \text{ if } K(\tilde{x}) := P$$

$$\text{(Par)} \quad \frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q} \qquad \text{(Res)} \quad \frac{P \rightarrow P'}{\nu a.P \rightarrow \nu a.P'} \qquad \text{(Struct)} \quad \frac{Q \equiv P \rightarrow P' \equiv Q'}{Q \rightarrow Q'}$$

Table 1. Rules defining the reaction relation $\rightarrow\ \subseteq \mathcal{P} \times \mathcal{P}$.

the reaction relation. The *transition system* of process P factorises the reachable processes along structural congruence, $\mathcal{T}(P) := (\mathit{Reach}(P)/_{\equiv}, \rightarrow_{\mathcal{T}}, [P])$ with the transition relation $[P] \rightarrow_{\mathcal{T}} [Q]$ defined by $P \rightarrow Q$.

Our theory employs two normal forms for processes. The classical *standard form* of Milner [Mil99] maximises the scopes of active restricted names and removes unused restrictions and $\mathbf{0}$ processes. For example, the standard form of $\nu a.K\lfloor a\rfloor \mid \nu b.0 \mid K\lfloor c\rfloor$ is $\nu a.(K\lfloor a\rfloor \mid K\lfloor c\rfloor)$. We compute it with the function sf, which is the identity on sequential processes. For a restriction $\nu a.P$ we have $\mathit{sf}(\nu a.P) := \nu a.\mathit{sf}(P)$ if $a \in \mathit{fn}(P)$ and $\mathit{sf}(\nu a.P) := \mathit{sf}(P)$ otherwise. The parallel composition of P and Q with $\mathit{sf}(P) = \nu\tilde{a}_P.P^{\neq\nu}$ and $\mathit{sf}(Q) = \nu\tilde{a}_Q.Q^{\neq\nu}$ maximises the scopes, $\mathit{sf}(P \mid Q) := \nu\tilde{a}_P.\nu\tilde{a}_Q.(P^{\neq\nu} \mid Q^{\neq\nu})$. Of course, the sequences of names $\tilde{a}_P$ and $\tilde{a}_Q$ may be empty and furthermore $P^{\neq\nu}$ and $Q^{\neq\nu}$ denote parallel compositions of sequential processes $K\lfloor a\rfloor$ and $M^{\neq\mathbf{0}}$.

Dual to the standard form, the *restricted form* [Mey09] minimises the scopes of active restrictions and also removes unused restrictions and processes congruent to $\mathbf{0}$. For example, the restricted form of $\nu a.(K\lfloor a\rfloor \mid \nu b.0 \mid K\lfloor c\rfloor)$ is $\nu a.K\lfloor a\rfloor \mid K\lfloor c\rfloor$. Technically, the restricted form relies on the notion of *fragments* F, G, H built inductively in two steps. Sequential processes $K\lfloor\tilde{a}\rfloor$ and $M^{\neq\mathbf{0}}$ are called *elementary fragments* F^e, G^e and form the basis. General fragments are defined by the grammar

$$F^e ::= K\lfloor\tilde{a}\rfloor \mid M^{\neq\mathbf{0}} \qquad F ::= F^e \mid \nu a.(F_1 \mid \ldots \mid F_n)$$

with $a \in \mathit{fn}(F_i)$ for all i. A *process* P^{rf} *in restricted form* is now a parallel compositions of fragments, $P^{rf} = \Pi_{i\in I}F_i$, and we refer to the fragments in P^{rf} by $\mathit{fg}(P^{rf}) := \bigcup_{i\in I}\{F_i\}$. The *decomposition function* $\mathit{dec}(P^{rf})$ counts the number of fragments in P^{rf} that are in a given class, e.g. $P^{rf} = F \mid G \mid F'$ with $F \equiv F' \not\equiv G$ yields $(dec(P^{rf}))([F]) = 2$, $(dec(P^{rf}))([G]) = 1$, and $(dec(P^{rf}))([H]) = 0$ for $F \not\equiv H \not\equiv G$. The function characterises structural congruence: $P^{rf} \equiv Q^{rf}$ is equivalent to $dec(P^{rf}) = dec(Q^{rf})$ [Mey09]. This is crucial for the well-definedness of the concurrency semantics we investigate in Section 3.

For a fragment F, we let $\|F\|_{\nu}$ denote the *nesting of active restrictions.* For example, $\|\nu a.(\nu b.K\lfloor a, b\rfloor \mid \nu c.\nu d.L\lfloor a, c, d\rfloor)\|_{\nu} = 3$. The *depth of a fragment* F is then the nesting of active restrictions in the flattest representation, $\|F\|_{\mathcal{D}} := \min\{\|G\|_{\nu} \mid G \equiv F\}$. A process is *bounded in depth* if there is a bound $k \in \mathbb{N}$ on the depth of all reachable fragments [Mey08a].

For a given process, function rf computes a structurally congruent one in restricted form [Mey09]. It is the identity on sequential processes and a homomorphism for the the parallel composition, $rf(P \mid Q) := rf(P) \mid rf(Q)$. In case of restriction $\nu a.P$, we first compute the restricted form $rf(P) = \Pi_{i \in I} F_i$. Then the scope of νa is restricted to the fragments where a is a free name (let the set I_a contain their indices), $rf(\nu a.P) := \nu a.(\Pi_{i \in I_a} F_i) \mid \Pi_{i \in I \setminus I_a} F_i$.

Petri Nets An *(unmarked) Petri net* is a triple (S, T, W) with disjoint and potentially infinite sets of *places* S and *transitions* T, and a *weight function* $W : (S \times T) \cup (T \times S) \to \mathbb{N} := \{0, 1, 2, \dots\}$. A Petri net is *finite* if S and T are finite sets. A *marking* of the net is a function $M : S \to \mathbb{N}$ and its *support* $supp(M)$ are the elements mapped to a value greater zero. As usual, places are represented by circles, transitions by boxes, the weight function by arcs with numbers (missing arcs have weight 0, unlabelled arcs have weight 1), and markings by tokens within the circles. We denote *pre-* and *postset* of $z \in S \cup T$ by ${}^{\bullet}z := \{y \mid W(y, z) > 0\}$ and $z^{\bullet} := \{y \mid W(z, y) > 0\}$. A *(marked) Petri net* is a pair $N = (S, T, W, M_0)$ of an unmarked net (S, T, W) and an *initial* marking M_0. The *set of all marked Petri nets* is denoted by $\mathcal{PN}$.

A transition $t \in T$ is *enabled* in marking M if $M(s) \geq W(s, t)$ for every $s \in {}^{\bullet}t$. *Firing* an enabled transition leads to marking $M'(s) := M(s) - W(s, t) + W(t, s)$ for every $s \in S$, and is denoted by $M[t\rangle M'$. In the transition system we work with the unlabelled relation $M \to M'$, which means $M[t\rangle M'$ for some $t \in T$.

To relate a process and its Petri net representation, we rely on the notion of bisimilarity [Mil89]. Two transition systems $\mathcal{T}_i = (St_i, \to_i, s_i^0)$ with $i = 1, 2$ are *bisimilar*, $\mathcal{T}_1 \approx \mathcal{T}_2$, if there is a bisimulation relation $\mathcal{R} \subseteq St_1 \times St_2$ that contains the initial states, $(s_1^0, s_2^0) \in \mathcal{R}$. In a *bisimulation relation* $\mathcal{R}$, containment $(s_1, s_2) \in \mathcal{R}$ requires (1) that for every $t_1 \in St_1$ with $s_1 \to_1 t_1$ there is a $t_2 \in St_2$ with $s_2 \to_2 t_2$ and $(t_1, t_2) \in \mathcal{R}$ and (2) similar for transitions from s_2.

Structural Semantics We recall the translation of π-calculus processes into Petri nets defined in [Mey09]. The idea is to have a separate place for each reachable group of processes connected by restricted names, i.e., the notion of fragments plays a crucial role. The algorithm takes a π-calculus process P and computes a Petri net $\mathcal{N}_{\mathcal{S}}[\![P]\!]$, called the *structural semantics* of P, as follows.

The places are the fragments of all reachable processes Q. More precisely, we take the structural congruence classes of fragments, $fg(rf(Q))/_{\equiv}$.

There are two disjoint sets of transitions. Transitions $t = ([F], [Q])$ model reactions inside fragment F that lead to process Q. More formally, $[F]$ is a place and $F \rightarrow Q$ holds. These reactions are communications within F, silent actions, or calls to process identifiers. There is an arc weighted one from place $[F]$ to t.

Transitions $t = ([F_1 \mid F_2], [Q])$ model reactions between reachable fragments along public channels: $[F_1]$ and $[F_2]$ are places and $F_1 \mid F_2 \rightarrow Q$. To define the preset of t, consider place $[F]$. If F_1, F_2, and F are structurally congruent there is an arc weighted two from $[F]$ to t. If F is structurally congruent with either F_1 or F_2, there is an arc weighted one. Otherwise there is no arc.

The postset of a transition $([F], [Q])$ or $([F_1 \mid F_2], [Q])$ are the reachable fragments of Q. If fragment G occurs (up to $\equiv$) $k \in \mathbb{N}$ times in $rf(Q)$, then there is an arc weighted k from $([F], [Q])$ to $[G]$. For example, from the transition $([\tau.(K\lfloor a\rfloor \mid K\lfloor a\rfloor)], [K\lfloor a\rfloor \mid K\lfloor a\rfloor])$ there is an arc weighted two to place $[K\lfloor a\rfloor]$.

The initial marking of place $[F]$ in $\mathcal{N}_{\mathcal{S}}[\![P]\!]$ is determined by the number of fragments in $rf(P)$ that are congruent to F. We illustrate the translation on our running example: a model of a bag data structure that takes values from a fill process on channel in and emits them on channel out in any order [Fok07].

Example 1. Consider $FILL\lfloor in\rfloor \mid BAG\lfloor in, out\rfloor$ with the equations $FILL(in) := \nu val.\overline{in}\langle val\rangle.FILL\lfloor in\rfloor$ and $BAG(in, out) := in(y).(\overline{out}\langle y\rangle \mid BAG\lfloor in, out\rfloor)$. The structural semantics $\mathcal{N}_{\mathcal{S}}[\![FILL\lfloor in\rfloor \mid BAG\lfloor in, out\rfloor]\!]$ is the Petri net in Figure 2 without $s_6, \ldots, s_9$ and with the following places:

$$s_1 := [FILL\lfloor in\rfloor] \qquad s_2 := [BAG\lfloor in, out\rfloor] \qquad s_5 := [\nu val.\overline{out}\langle val\rangle]$$
$$s_3 := [\nu val.\overline{in}\langle val\rangle.FILL\lfloor in\rfloor] \qquad s_4 := [in(y).(\overline{out}\langle y\rangle \mid BAG\lfloor in, out\rfloor)].$$

Note that the structural semantics remains finite if we restrict the name in while it becomes infinite if we restrict out. A restricted name out, distributed to an unbounded number of processes $\nu val.\overline{out}\langle val\rangle$, would create an unbounded number of places (unboundedness in breadth [Mey09]).

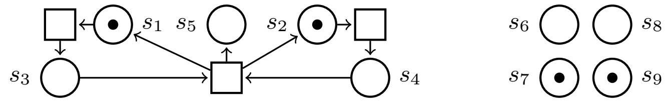

Fig. 2. Petri net representation of the different versions of the bag data structure introduced in the Examples 1, 2, and 3. The places and the usage of $s_6, \ldots, s_9$ depend on the semantics under consideration and are explained in the examples.

A π-calculus process and its structural semantics have isomorphic transition systems [Mey09]. Moreover, the structural semantics is finite iff the translated process is structurally stationary, i.e., generates finitely many types of fragments.

3 A Sound Concurrency Semantics for the π-calculus

Concurrency semantics highlight the communications between sequential processes. As opposed to the structural semantics, the scopes of restricted names are not important. The key idea to define a concurrency semantics is to use designated free names for active restrictions. To generate these names systematically, we define the so-called *name-aware transition system*. It facilitates the first bisimilarity proof $P \approx \mathcal{N}_{\mathcal{C}}[\![P]\!]$ for a concurrency semantics that handles name creation and allows for a finiteness characterisation. Surprisingly, all earlier attempts lacking this technical tool failed as explained in the introduction.

For a smooth definition of the name-aware transition system we assume that restricted names have the form a_m, i.e., they carry an *index* $m \in \mathbb{N}$. α-conversion of a restricted name a_m changes the index m but not the name a. Wlog., for a process P of interest we assume the indices to be zero. If P uses defining equations $K_i(\tilde{x}) := P_i$, then the indices in the P_i are zero as well. For a restriction a_m the *increment operation* yields $a_m + 1 := a_{m+1}$. Its application to sets is defined elementwise, for example $\{a_3, b_2, c_5\} + 1 = \{a_4, b_3, c_6\}$.

In the name-aware transition system, the states are *name-aware processes* of the form $(P^{\neq\nu}, \tilde{a})$. Intuitively, in an execution sequence leading to process $(P^{\neq\nu}, \tilde{a})$ the active restrictions $\tilde{a}$ have been found (the set may be empty, $\tilde{a} = \emptyset$). The names $\tilde{a}$ are not chosen arbitrarily but are computed by incrementing the indices. For example, the name-aware process $(\tau.\nu a_0.K\lfloor a_0\rfloor, \{a_0, a_1, a_2\})$ consumes a τ-action and generates the restricted name a_3. Formally, the behaviour of name-aware processes is captured by the *name-aware reaction relation*

$$(P^{\neq\nu}, \tilde{a}) \rightarrow^{na} (Q^{\neq\nu}, \tilde{a} \uplus \tilde{b}) :\Leftrightarrow \begin{array}{l} P^{\neq\nu} \rightarrow \nu\tilde{b}.Q^{\neq\nu} \text{ in standard form and} \\ \forall b_k \in \tilde{b} : k - 1 = max\{i \mid b_i \in \tilde{a}\}, \end{array} \qquad (\clubsuit)$$

where we choose zero as index if there is no name $b_i \in \tilde{a}$. The set of all processes reachable from $(P^{\neq\nu}, \tilde{a})$ by the name-aware reaction relation is $Reach_{na}(P^{\neq\nu}, \tilde{a})$. For the example process above, the definition in fact yields the name-aware reaction $(\tau.\nu a_0.K\lfloor a_0\rfloor, \{a_0, a_1, a_2\}) \rightarrow^{na} (K\lfloor a_3\rfloor, \{a_0, a_1, a_2, a_3\})$. The *name-aware transition system* $\mathcal{T}_{na}(P^{\neq\nu}, \tilde{a})$ is again defined by factorising the reachable processes along structural congruence, $Reach_{na}(P^{\neq\nu}, \tilde{a})/_{\equiv}$. The name-aware reaction relation is lifted to process classes $([P^{\neq\nu}], \tilde{a})$, accordingly. Lemma 1 states bisimilarity of the name-aware and the standard transition system of a process.

Lemma 1. *For every process $P \in \mathcal{P}$ with standard form $sf(P) = \nu\tilde{a}.P^{\neq\nu}$ the bisimilarity $\mathcal{T}(P) \approx \mathcal{T}_{na}(P^{\neq\nu}, \tilde{a})$ holds.*

Proof. The relation that connects

$$([Q^{\neq\nu}], \tilde{b}) \in Reach_{na}(P^{\neq\nu}, \tilde{a})/_{\equiv} \qquad \text{and} \qquad [\nu\tilde{b}.Q^{\neq\nu}] \in Reach(P)/_{\equiv}$$

is a bisimulation and relates the initial processes in both transition systems.

Two basic ideas underly our concurrency semantics. First, as usual we let tokens reflect the numbers of processes in a state. Second and unusual, we use additional *name places* to imitate the generation of free names in the name-aware transition system; a construction that has precursor in our earlier works [BG95,BG09]. If a place, say a_3, is marked, the names a_0, a_1, a_2 have already been generated and a_3 is the next one to be invented. The transition that corresponds to the reaction $(\tau.\nu a_0.K\lfloor a_0\rfloor, \{a_0, a_1, a_2\}) \rightarrow^{na} (K\lfloor a_3\rfloor, \{a_0, a_1, a_2, a_3\})$ above moves the token from name place a_3 to a_4.

Technically, we start with the name-aware transition system and compute the two disjoint sets of *name* and *process places*. The name places are precisely the names $\tilde{b}$ in all reachable name-aware processes $([Q^{\neq\nu}], \tilde{b})$. The process places are given by the (structural congruence classes of) sequential processes in $Q^{\neq\nu}$.

Let $([P^{\neq\nu}], \tilde{a})$ be the initial process in the name-aware transition system. Also the initial marking is composed out of two disjoint functions. Function $M_0^{\mathcal{P}}$ marks the process places as required by the sequential processes in $P^{\neq\nu}$. Marking $M_0^{\mathcal{N}}$ puts a single token on all name places with index zero—except $\tilde{a}$. If $a_0 \in \tilde{a}$ the name is already in use and a_1 is the next to be generated. Therefore, name place a_1 is marked by one token. In fact, all name places are 1-safe.

Like for the structural semantics we have two disjoint sets of transitions. The first set contains transitions $t = ([M^{\neq\mathbf{0}}], \tilde{b}, [Q^{\neq\nu}])$ with the constraint that $[M^{\neq\mathbf{0}}]$ and $\tilde{b}$ are places and $M^{\neq\mathbf{0}} \rightarrow \nu\tilde{b}.Q^{\neq\nu}$ in standard form. The preset of t are the process place $[M^{\neq\mathbf{0}}]$ and the name places $\tilde{b}$. Hence, names can only be generated if their places are marked. The postset is given by the reachable sequential processes in $Q^{\neq\nu}$ and the names $\tilde{b}+1$. Thus, the transition moves a token from $b_k \in \tilde{b}$ to b_{k+1} as was explained. Similar transitions exist for $K\lfloor\tilde{c}\rfloor$.

The second set of transitions models communications between sequential processes. Here we have transitions $t = ([M_1^{\neq\mathbf{0}} \mid M_2^{\neq\mathbf{0}}], \tilde{b}, [Q^{\neq\nu}])$ with the condition that $M_1^{\neq\mathbf{0}} \mid M_2^{\neq\mathbf{0}}$ reacts to $\nu\tilde{b}.Q^{\neq\nu}$ in standard form. There is an arc weighted two from place $[N^{\neq\mathbf{0}}]$ to t if $M_1^{\neq\mathbf{0}} \equiv N^{\neq\mathbf{0}} \equiv M_2^{\neq\mathbf{0}}$. In this case, two structurally congruent processes communicate. If place $[N^{\neq\mathbf{0}}]$ is only one of the sequential processes, $N^{\neq\mathbf{0}} \equiv M_1^{\neq\mathbf{0}}$ xor $N^{\neq\mathbf{0}} \equiv M_2^{\neq\mathbf{0}}$, we draw an arc weighted one from the place to the transition. In any other case there is no arc, which means transition t represents a reaction where process $[N^{\neq\mathbf{0}}]$ is not involved in. Like for the first set of transitions, the places $\tilde{b}$ in the preset of t ensure restricted names are invented in the correct order. The postset is similar as well. We illustrate the definition of the concurrency semantics on the bag data structure.

Example 2. Consider the process $\nu in_0.\nu out_0.(FILL_2\lfloor in_0, val\rfloor \mid BAG\lfloor in_0, out_0\rfloor)$ where different from Example 1 data value val is not restricted, $FILL_2(in_0, val) := \overline{in_0}\langle val\rangle.FILL_2\lfloor in_0, val\rfloor$. The equation for BAG is as before. The concurrency semantics $\mathcal{N}_{\mathcal{C}}[\![\nu in_0.\nu out_0.(FILL_2\lfloor in_0, val\rfloor \mid BAG\lfloor in_0, out_0\rfloor)]\!]$ is the Petri net in Figure 2 with the process places

$$s_1 := [FILL_2\lfloor in_0, val\rfloor] \qquad s_2 := [BAG\lfloor in_0, out_0\rfloor] \qquad s_5 := [\overline{out_0}\langle val\rangle]$$
$$s_3 := [\overline{in_0}\langle val\rangle.FILL_2\lfloor in_0, val\rfloor] \qquad s_4 := [in_0(y).(\overline{out_0}\langle y\rangle \mid BAG\lfloor in_0, out_0\rfloor)]$$

and the name places $s_6 := in_0$, $s_7 := in_1$, $s_8 := out_0$, and $s_9 := out_1$. Note that place in_1 is initially marked as in_0 is active in the initial process. Moreover, if we restricted *val* the concurrency semantics would have an unbounded number of val_i places and hence be infinite.

Formally, the *concurrency semantics* is the function $\mathcal{N}_{\mathcal{C}} : \mathcal{P} \to \mathcal{PN}$. It assigns to a process $P_0 \in \mathcal{P}$ with $sf(P_0) = \nu\tilde{a}_0.P_0^{\neq\nu}$ the Petri net $\mathcal{N}_{\mathcal{C}}[\![P_0]\!] = (S, T, W, M_0)$ as follows. The set of places is the disjoint union $S := S^{\mathcal{P}} \uplus S^{\mathcal{N}}$ of the process places $S^{\mathcal{P}}$ and the name places $S^{\mathcal{N}}$:

$$S^{\mathcal{P}} := fg(Reach_{na}(P_0^{\neq\nu}, \tilde{a}_0))/_{\equiv}$$
$$S^{\mathcal{N}} := nms(Reach_{na}(P_0^{\neq\nu}, \tilde{a}_0)) \cup nms(Reach_{na}(P_0^{\neq\nu}, \tilde{a}_0)) + 1,$$

where $fg(P^{\neq\nu}, \tilde{a}) := fg(P^{\neq\nu})$ and $nms(P^{\neq\nu}, \tilde{a}) := \tilde{a}$ for a name-aware process $(P^{\neq\nu}, \tilde{a})$. Note that since $P^{\neq\nu}$ is a parallel composition of elementary fragments, it is in restricted form and we can access its elements via $fg(P^{\neq\nu})$.

To define the set T of transitions, consider the process places $[F^e]$, $[F_1^e]$, $[F_2^e]$ and the name places $\tilde{a}$ in S. We have

$$\begin{aligned} T := \quad & \{([F^e], \tilde{a}, [Q^{\neq\nu}]) \mid F^e \to \nu\tilde{a}.Q^{\neq\nu} \text{ in standard form}\} \\ \cup\ & \{([F_1^e \mid F_2^e], \tilde{a}, [Q^{\neq\nu}]) \mid F_1^e \mid F_2^e \to \nu\tilde{a}.Q^{\neq\nu} \text{ in standard form}\}. \end{aligned}$$

We define the weight function for transitions $t = ([F_1^e \mid F_2^e], \tilde{a}, [Q^{\neq\nu}])$, for transitions $t' = ([F^e], \tilde{a}, [Q^{\neq\nu}])$ the definition is similar. Consider places a and $[G^e]$ and let condition $a \in \tilde{a}$ or $a \in (\tilde{a} + 1)$ yield 1 if it is satisfied and 0 otherwise:

$$\begin{aligned} W([G^e], t) &:= (dec(F_1^e \mid F_2^e))([G^e]) \qquad & W(a, t) &:= a \in \tilde{a} \\ W(t, [G^e]) &:= (dec(Q^{\neq\nu}))([G^e]) \qquad & W(t, a) &:= a \in (\tilde{a} + 1). \end{aligned}$$

The initial marking is the disjoint union $M_0 := M_0^{\mathcal{P}} \uplus M_0^{\mathcal{N}}$. Since name places receive a single token, we define $M_0^{\mathcal{N}}$ by the set of marked places:

$$M_0^{\mathcal{P}} := dec(P_0^{\neq\nu}) \qquad M_0^{\mathcal{N}} := (\{a_0 \in S\} \setminus \tilde{a}_0) \cup (\tilde{a}_0 + 1).$$

This definition in fact mirrors the name-aware transition system.

Lemma 2. *For every process $P \in \mathcal{P}$ with $sf(P) = \nu\tilde{a}.P^{\neq\nu}$ the bisimilarity $\mathcal{T}(\mathcal{N}_{\mathcal{C}}[\![P]\!]) \approx \mathcal{T}_{na}(P^{\neq\nu}, \tilde{a})$ holds.*

Proof. The lemma can be established by showing that

$$\begin{aligned} \mathcal{R} := \big\{\big(M^{\mathcal{P}} \uplus M^{\mathcal{N}}, ([Q^{\neq\nu}], \tilde{b})\big) \mid Q^{\neq\nu} \equiv \mathit{\Pi}_{[F^e] \in supp(M^{\mathcal{P}})} \mathit{\Pi}^{M^{\mathcal{P}}([F^e])} F^e \text{ and} \\ \tilde{b} = \{b_i \in S \mid M^{\mathcal{N}}(b_k) = 1 \text{ with } i < k\}\big\} \end{aligned}$$

is a bisimulation relation that connects the initial state of the name-aware transition system and the initial marking of the concurrency semantics.

By transitivity of bisimilarity, the Lemmas 1 and 2 prove our first main result. The transition system of a process and that of its concurrency semantics are bisimilar and the reachable processes can be recomputed from the markings using the composed bisimulation relation.

Theorem 1 (Retrievability). *For every $P \in \mathcal{P}$ we have $\mathcal{T}(\mathcal{N}_{\mathcal{C}}[\![P]\!]) \approx \mathcal{T}(P)$.*

Our second result is a finiteness characterisation. The concurrency semantics is a finite Petri net if and only if the process generates finitely many restricted names. We call those processes *restriction bounded.* Since in the standard transition system unused restrictions can be removed by $\nu a.P \equiv P$ if $a \notin fn(P)$, we again rely on the name-aware transition system to define restriction boundedness.

Definition 1. *Consider $P \in \mathcal{P}$ with $sf(P) = \nu\tilde{a}.P^{\neq\nu}$. We call P* restriction bounded *if there is a finite set of names $\tilde{m} \subseteq \mathcal{N}$ so that for every reachable name-aware process $(Q^{\neq\nu}, \tilde{b})$ the inclusion $\tilde{b} \subseteq \tilde{m}$ holds.*

If the process is not restriction bounded, clearly the concurrency semantics is infinite as every restricted name yields a place. Theorem 2 shows that also the reverse holds, i.e., a bounded number of restricted names implies finiteness of the Petri net. The proof uses the theory of derivatives [Mey09, Proposition 3].

Theorem 2 (Finiteness Characterisation). *For any process $P \in \mathcal{P}$, the concurrency semantics $\mathcal{N}_{\mathcal{C}}[\![P]\!]$ is finite if and only if P is restriction bounded.*

The Examples 1 and 2 show that structurally stationary and restriction-bounded processes are incomparable. Section 4 explains how to unify the classes.

4 Combining Structural and Concurrency Semantics

To combine the structural and the concurrency semantics we type restricted names, and the type determines the semantics that handles a name. More precisely, restricted names νa may carry a tag $\mathcal{C}$. Tagged names $\nu a^{\mathcal{C}}$ are translated by the concurrency semantics while names νa without tag are handled by the structural semantics. Hence, tagged names yield name places in the combined semantics and untagged names form fragments that replace the process places. Like in the concurrency semantics, we assume that tagged names have indices.

Technically, the idea of adding tags raises two problems that need to be addressed. (1) We need to define a name-aware transition system to generate tagged names systematically. (2) We need to compute the fragments formed by untagged names. The solution to both problems is a normal form for processes, which combines the standard and the restricted form. Before we turn to its definition, we illustrate the use of tags on our running example.

Example 3. Tagging the names in_0 and out_0 in the bag example yields process $\nu in_0^{\mathcal{C}}.\nu out_0^{\mathcal{C}}.(FILL\lfloor in_0^{\mathcal{C}}\rfloor \mid BAG\lfloor in_0^{\mathcal{C}}, out_0^{\mathcal{C}}\rfloor)$ with the equations in Example 1. Since channel $out_0^{\mathcal{C}}$ is shared by arbitrarily many processes, we tag it to treat it by the concurrency semantics. Vice versa, as arbitrarily many instances of val are created we omit the tag to handle the name by the structural semantics.

The mixed translation will result in the Petri net in Figure 2 with the name places $s_6 := in_0^{\mathcal{C}}$, $s_7 := in_1^{\mathcal{C}}$, $s_8 := out_0^{\mathcal{C}}$, and $s_9 := out_1^{\mathcal{C}}$. Different from the concurrency semantics, the mixed semantics has *fragment places*

$$s_1 := [FILL\lfloor in_0^{\mathcal{C}}\rfloor] \qquad s_2 := [BAG\lfloor in_0^{\mathcal{C}}, out_0^{\mathcal{C}}\rfloor] \qquad s_5 := [\nu val.\overline{out_0^{\mathcal{C}}}\langle val\rangle]$$
$$s_3 := [\nu val.\overline{in_0^{\mathcal{C}}}\langle val\rangle.FILL\lfloor in_0^{\mathcal{C}}\rfloor] \qquad s_4 := [in_0^{\mathcal{C}}(y).(\overline{out_0^{\mathcal{C}}}\langle y\rangle \mid BAG\lfloor in_0^{\mathcal{C}}, out_0^{\mathcal{C}}\rfloor)].$$

Observe that neither the structural nor the concurrency semantics finitely represent the process. It is also worth comparing the places with those in the Examples 1 and 2 to see how the mixed semantics unifies the previous translations.

4.1 Mixed Normal Form

The idea of the mixed normal form is to *maximise* the scopes of tagged active restrictions $\nu a^{\mathcal{C}}$ and to *minimise* the scopes of untagged ones νa. In the resulting process $P^{mf} = \nu\tilde{a}^{\mathcal{C}}.P^{rf}$ the tagged names $\nu\tilde{a}^{\mathcal{C}}$ surround a process P^{rf} in restricted form, which only contains untagged active restrictions. We call P^{mf} a process *in mixed normal form* and denote the *set of all processes in mixed normal form* by $\mathcal{P}_{mf}$. To give an example, process

$$P = \nu in_0^{\mathcal{C}}.\nu out_0^{\mathcal{C}}.(FILL\lfloor in_0^{\mathcal{C}}\rfloor \mid BAG\lfloor in_0^{\mathcal{C}}, out_0^{\mathcal{C}}\rfloor \mid \overline{out_0^{\mathcal{C}}}\langle val\rangle)$$

is in mixed normal form while $\nu val.P$ is not as the scope of νval is not minimal.

For every tagged process, the function $mf : \mathcal{P} \to \mathcal{P}_{mf}$ computes a structurally congruent process in mixed normal form. Empty sums $M^{=\mathbf{0}}$ are mapped to $\mathbf{0}$ and sequential processes are left unchanged. For the parallel composition $P \mid Q$, we recursively compute the mixed normal forms $mf(P) = \nu\tilde{a}_P^{\mathcal{C}}.P^{rf}$ and $mf(Q) = \nu\tilde{a}_Q^{\mathcal{C}}.Q^{rf}$, where $\tilde{a}_P^{\mathcal{C}}$ and $\tilde{a}_Q^{\mathcal{C}}$ may be empty. Like the standard form, the mixed normal form extrudes the scopes of $\tilde{a}_P^{\mathcal{C}}$ and $\tilde{a}_Q^{\mathcal{C}}$,

$$mf(P \mid Q) := \nu\tilde{a}_P^{\mathcal{C}}.\nu\tilde{a}_Q^{\mathcal{C}}.(P^{rf} \mid Q^{rf}).$$

Tagged active restricted names are handled like in the standard form, i.e., we have $mf(\nu a^{\mathcal{C}}.P) := \nu a^{\mathcal{C}}.mf(P)$ if $a^{\mathcal{C}} \in fn(P)$ and $mf(\nu a^{\mathcal{C}}.P) := mf(P)$ otherwise. For a process $\nu a.P$ with an untagged name νa, we recursively compute $mf(P) = \nu\tilde{a}^{\mathcal{C}}.P^{rf}$. This singles out the tagged names $\tilde{a}^{\mathcal{C}}$ in P. Commuting νa with $\nu\tilde{a}^{\mathcal{C}}$ yields $\nu\tilde{a}^{\mathcal{C}}.\nu a.P^{rf}$. Let $P^{rf} = \Pi_{i\in I}F_i$ and let I_a contain the indices of the fragments that have a as a free name. Shrinking the scope of νa gives

$$mf(\nu a.P) := \nu\tilde{a}^{\mathcal{C}}.\left(\nu a.(\Pi_{i\in I_a}F_i) \mid \Pi_{i\in I\setminus I_a}F_i\right).$$

Example 4. We observed that $\nu val.P \notin \mathcal{P}_{mf}$. An application of the function gives $mf(\nu val.P) = \nu in_0^{\mathcal{C}}.\nu out_0^{\mathcal{C}}.(\nu val.\overline{out_0^{\mathcal{C}}}\langle val\rangle \mid FILL\lfloor in_0^{\mathcal{C}}\rfloor \mid BAG\lfloor in_0^{\mathcal{C}}, out_0^{\mathcal{C}}\rfloor)$, which is a process in mixed normal form.

Lemma 3. *For every tagged process $P \in \mathcal{P}$ function mf yields $mf(P) \in \mathcal{P}_{mf}$ with $mf(P) \equiv P$. For $P^{mf} \in \mathcal{P}_{mf}$, even $mf(P^{mf}) = P^{mf}$ holds.*

4.2 Mixed Semantics

Like the concurrency semantics, the definition of the mixed semantics relies on a name-aware transition system that organises the creation of fresh tagged active restrictions. With the mixed normal form $\nu\tilde{a}^{\mathcal{C}}.P^{rf}$ in mind, we define the new *name-aware processes* to be pairs $(P^{rf}, \tilde{a}^{\mathcal{C}})$, where the active restrictions in P^{rf} are untagged. For these name-aware processes, the *name-aware reaction relation* $\rightarrow^{na}$ is adapted accordingly. The only difference to Definition (♣) is the use of the mixed normal form instead of the standard form:

$$(P^{rf}, \tilde{a}^{\mathcal{C}}) \rightarrow^{na} (Q^{rf}, \tilde{a}^{\mathcal{C}} \uplus \tilde{b}^{\mathcal{C}}) :\Leftrightarrow \begin{array}{l} P^{rf} \rightarrow \nu\tilde{b}^{\mathcal{C}}.Q^{rf} \text{ in mixed normal form and} \\ \forall b_k^{\mathcal{C}} \in \tilde{b}^{\mathcal{C}} : k-1 = max\{i \mid b_i^{\mathcal{C}} \in \tilde{a}^{\mathcal{C}}\}. \end{array}$$

Without change of notation, let the resulting new name-aware transition system be $\mathcal{T}_{na}(P^{rf}, \tilde{a}^{\mathcal{C}})$. It is straightforward to modify the proof of Lemma 1 in order to show bisimilarity for the adjusted definition.

Lemma 4. *For every tagged process $P \in \mathcal{P}$ with $mf(P) = \nu\tilde{a}^{\mathcal{C}}.P^{rf}$ the bisimilarity $\mathcal{T}_{na}(P^{rf}, \tilde{a}^{\mathcal{C}}) \approx \mathcal{T}(P)$ holds.*

The mixed semantics is a variant of the concurrency semantics, so again we have two sets of places. While the tagged active restrictions $\tilde{b}^{\mathcal{C}}$ in name-aware processes $([Q^{rf}], \tilde{b}^{\mathcal{C}})$ yield *name places*, the process places known from the concurrency semantics are replaced by *fragment places* $fg(Q^{rf})/_{\equiv}$ in the mixed semantics. They represent the fragments generated by untagged active restrictions. Also the transitions are changed to the form

$$t = ([F], \tilde{b}^{\mathcal{C}}, [Q^{rf}]) \quad \text{and} \quad t = ([F_1 \mid F_2], \tilde{b}^{\mathcal{C}}, [Q^{rf}]),$$

with the condition that F (and $F_1 \mid F_2$) has a name-aware reaction to the process $\nu\tilde{b}^{\mathcal{C}}.Q^{rf}$ in mixed normal form, i.e., $F \rightarrow^{na} \nu\tilde{b}^{\mathcal{C}}.Q^{rf}$ and $\nu\tilde{b}^{\mathcal{C}}.Q^{rf} \in \mathcal{P}_{mf}$. Intuitively, the transition models a communication of processes (or a τ-action or a call to an identifier) within F, which results in process Q^{rf} and generates the new tagged active restrictions $\tilde{b}^{\mathcal{C}}$. The modification of the weight function is immediate. We denote the *mixed semantics* of a process $P \in \mathcal{P}$ by $\mathcal{N}_{\mathcal{M}}[\![P]\!]$. Also the proof of bisimilarity in Lemma 2 still holds for the mixed semantics.

Lemma 5. *Consider the tagged process $P \in \mathcal{P}$ with $mf(P) = \nu\tilde{a}^{\mathcal{C}}.P^{rf}$. We have the bisimilarity $\mathcal{T}(\mathcal{N}_{\mathcal{M}}[\![P]\!]) \approx \mathcal{T}_{na}(P^{rf}, \tilde{a}^{\mathcal{C}})$.*

Combining the bisimilarities in Lemma 4 and 5, we obtain bisimilarity for the mixed semantics. Moreover, the bisimulation relation used in the proof allows one to reconstruct the reachable process terms from the markings.

Theorem 3 (Retrievability). *For every tagged process $P \in \mathcal{P}$ the bisimilarity $\mathcal{T}(\mathcal{N}_{\mathcal{M}}[\![P]\!]) \approx \mathcal{T}(P)$ holds.*

For processes *without tagged names* the mixed semantics degenerates to the structural semantics. The absence of tagged names leads to absence of name places in the semantics. Hence, transitions do not generate names and have the form $([F], \emptyset, [Q^{rf}])$ or $([F_1 \mid F_2], \emptyset, [Q^{rf}])$. They can be identified with the transitions $([F], [Q])$ and $([F_1 \mid F_2], [Q])$ in the structural semantics. In case *all names are tagged* in the process under consideration, the mixed semantics corresponds to the concurrency semantics. This follows from the fact that the mixed normal form coincides with the standard form for these processes. Hence, the fragment places in the mixed semantics are in fact sequential processes.

Proposition 1 (Conservative Extension). *If the process $P \in \mathcal{P}$ does not use tagged names, the mixed semantics coincides with the structural semantics, i.e., $\mathcal{N}_{\mathcal{M}}[\![P]\!] = \mathcal{N}_{\mathcal{S}}[\![P]\!]$. If the process only uses tagged names, the mixed semantics coincides with the concurrency semantics, i.e., $\mathcal{N}_{\mathcal{M}}[\![P]\!] = \mathcal{N}_{\mathcal{C}}[\![P]\!]$.*

According to Theorem 2 the concurrency semantics is finite if and only if the translated process is restriction bounded. The structural semantics is finite precisely if there is a finite set of fragments every reachable process consists of (structural stationarity [Mey09]). We prove the mixed semantics to be finite if and only if (a) finitely many tagged names are generated and (b) the untagged names form finitely many fragments.

Definition 2. *A tagged process $P \in \mathcal{P}$ is* mixed bounded *if there are finite sets of names $\tilde{m}^{\mathcal{C}}$ and fragments $\{F_1, \ldots, F_n\}$ so that for every reachable process $(Q^{rf}, \tilde{b}^{\mathcal{C}})$ we have $\tilde{b}^{\mathcal{C}} \subseteq \tilde{m}^{\mathcal{C}}$ and for every $F \in fg(Q^{rf})$ there is F_i with $F \equiv F_i$.*

To see that $\mathcal{N}_{\mathcal{M}}[\![P]\!]$ is finite iff P is mixed bounded, observe that finiteness of the set of places implies finiteness of the mixed semantics. Finiteness of the set of name places is equivalent to Condition (a), finiteness of the set of fragment places equivalent to Condition (b) in the definition of mixed boundedness.

Theorem 4 (Finiteness Characterisation). *For every tagged process $P \in \mathcal{P}$ the mixed semantics $\mathcal{N}_{\mathcal{M}}[\![P]\!]$ is finite if and only if P is mixed bounded.*

Structurally stationary and restriction-bounded processes are mixed bounded, hence the mixed semantics finitely represents all their *syntactic* subclasses. In *finite control processes* parallel compositions are forbidden within recursions, but an unbounded number of restricted names may be generated [Dam96]. Incomparable, *restriction-free processes* allow for unbounded parallelism but forbid the use of restrictions [AM02]. They are generalised by *finite handler processes* designed for modelling client-server systems [Mey09]. All these classes are structurally stationary [Mey09]. Restriction-free processes are also generalised by *finite-net processes*, which forbid the use of restrictions within recursions but allow for unbounded parallelism (dual to finite control processes) [BG09]. They form a subclass of restriction-bounded processes.

We conclude the section with a remark that mixed-bounded processes are a subclass of the processes of bounded depth.

Proposition 2. *If $P \in \mathcal{P}$ is mixed bounded, then it is bounded in depth.*

Theorem 4 shows that if a process is mixed bounded then *there is* a faithful representation as a finite p/t net. We now consider the reverse direction.

5 Borderline to Finite P/T Petri Nets

We argue that if we have a superclass of mixed-bounded processes, there will be no reachability-preserving translation into finite p/t Petri nets. This means mixed-bounded processes form the borderline between π-calculus and finite p/t nets. Since it is always possible to handle particular classes of processes by specialised translations, we make our argument precise. We show that in a *slight extension* of mixed-bounded processes reachability becomes undecidable. Since the problem is decidable for finite p/t nets [May84], there is no reachability-preserving translation for the extended process class.

The processes we consider are *bounded in depth by one*. To establish undecidability of reachability, we reduce the corresponding problem for 2-counter machines [Min67]. Due to the limitations of the process class, the counter machine encoding differs drastically from those in the literature [Mil89,AM02,BGZ03] modelling counters as stacks. The only related encoding that does not rely on stacks is given in [BGZ04] to prove weak bisimilarity undecidable for CCS$_!$.

We imitate a construction in [DFS98] which shows undecidability of reachability for transfer nets. The idea of Dufourd, Finkel, and Schnoebelen is to represent a counter c_1 by two places c_1 and c_1'. The test for zero

$$l : \text{if } c_1 = 0 \text{ then goto } l'; \text{ else } c_1 := c_1 - 1; \text{ goto } l''; \qquad (\spadesuit)$$

is modelled by the transfer net in Figure 3. To test counter c_1 for being zero, transition t transfers the content of c_1' to a trash place s_t. Since the transition is enabled although c_1' is empty, the content of c_1 and c_1' coincides as long as the net properly simulates the counter machine. If a transfer operation is executed when c_1' is not empty, the amount of tokens in c_1 and c_1' becomes different, Figure 3 (a) and (b). The difference is preserved throughout the computation, because increment operations add the same amount of tokens to c_1 and c_1'. Hence, a state $(c_1 = v_1, c_2 = v_2, l)$ with $v_1, v_2 \in \mathbb{N}$ is reachable in the counter machine if and only if a marking is reachable in the transfer net where place l is marked, counter c_1 and its copy c_1' carry v_1 tokens, and similar for c_2 and c_2'.

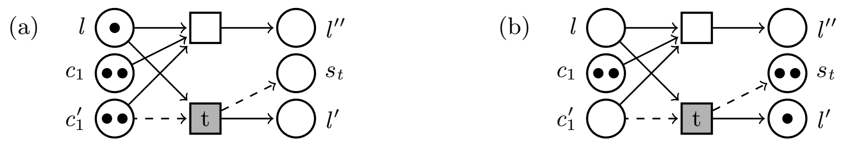

Fig. 3. A Petri net with transfer modelling a test for zero in a counter machine. Dashed lines represent transfer arcs of t that move all tokens in c_1' to the trash place s_t.

We represent a counter value by a parallel composition of processes, e.g. $c_1' = 2$ by $\overline{a} \mid \overline{a}$. The transfer operation requires us to change arbitrarily many processes with one communication. To achieve this, we attach the processes $\overline{a}$ to a so-called *process bunch* $PB\lfloor a, i_{c_1'}, d_{c_1'}, t_{c_1'}\rfloor$ by restricting the name a. The result is a process $\nu a.(PB\lfloor a, i_{c_1'}, d_{c_1'}, t_{c_1'}\rfloor \mid \overline{a} \mid \overline{a})$. Since the name a is restricted, the process bunch has exclusive access to its processes $\overline{a}$. It offers three operations to modify their numbers. A communication on $i_{c_1'}$ stands for *increment* and creates a new process $\overline{a}$. Similarly, a message on $d_{c_1'}$ *decrements* the process number by consuming a process $\overline{a}$. A *test for zero* on $t_{c_1'}$ creates a new and empty process bunch for counter c_1'. The old process bunch terminates. A process $\nu a.(\overline{a} \mid \overline{a})$ without process bunch is considered to belong to the trash place. Abbreviating d_x, i_x, t_x by $\tilde{c}_x$, a process bunch is defined by

$$PB(a, \tilde{c}_x) := i_x.(PB\lfloor a, \tilde{c}_x\rfloor \mid \overline{a}) + d_x.a.PB\lfloor a, \tilde{c}_x\rfloor + t_x.\nu b.PB\lfloor b, \tilde{c}_x\rfloor.$$

Labelled instructions $l : \mathit{inst}$ of the counter machine are translated to process identifiers K_l where *inst* determines the defining process. The increment operation $l : c_1 := c_1 + 1$ goto l' yields $K_l(\tilde{c}) := \overline{i_{c_1}}.\overline{i_{c_1'}}.K_{l'}\lfloor\tilde{c}\rfloor$. Here, $\tilde{c}$ abbreviates the channels of all four process bunches $d_{c_1}, i_{c_1}, t_{c_1}, d_{c_1'}, \ldots, t_{c_2'}$. Note that both, c_1 and c_1', are incremented. Like in transfer nets, the test for zero (♠) only changes the value of counter c_1'. Decrement acts on both counters:

$$K_l(\tilde{c}) := \overline{t_{c_1'}}.K_{l'}\lfloor\tilde{c}\rfloor + \overline{d_{c_1}}.\overline{d_{c_1'}}.K_{l''}\lfloor\tilde{c}\rfloor.$$

If an empty process bunch accepts a decrement, the system deadlocks (requires synchronisation actions that we omitted here to ease presentation) and reachability is preserved. Finally, a halt instruction $l : \mathit{halt}$ is translated into $K_l(\tilde{c}) := \overline{\mathit{halt}}$. The full translation of a counter machine CM yields the process

$$\mathcal{P}[\![CM]\!] := \mathit{\Pi}_{x \in \{c_1, \ldots, c_2'\}} \nu a_x.PB\lfloor a_x, \tilde{c}_x\rfloor \mid K_{l_0}\lfloor\tilde{c}\rfloor.$$

The counter machine CM reaches the state $(c_1 = v_1, c_2 = v_2, l)$ if and only if its encoding $\mathcal{P}[\![CM]\!]$ reaches the process

$$\mathit{\Pi}_{x \in \{c_1, c_1'\}} \nu a_x.(PB\lfloor a_x, \tilde{c}_x\rfloor \mid \mathit{\Pi}^{v_1}\overline{a_x}) \mid \mathit{\Pi}_{x \in \{c_2, c_2'\}} \nu a_x.(PB\lfloor a_x, \tilde{c}_x\rfloor \mid \mathit{\Pi}^{v_2}\overline{a_x}) \mid K_l\lfloor\tilde{c}\rfloor.$$

The leftmost parallel composition ensures that the process bunches for c_1 and c_1' contain v_1 processes $\overline{a_{c_1}}$ and $\overline{a_{c_1'}}$, respectively. The construction for c_2 and c_2' is similar. Combined with the observation that the process $\mathcal{P}[\![CM]\!]$ is bounded in depth by one, we arrive at the desired undecidability result.

Theorem 5 (Undecidability in Depth One). *Consider $P, Q \in \mathcal{P}$ where the depth is bounded by one. The problem whether $Q \in \mathit{Reach}(P)$ is undecidable.*

Since reachability is decidable for finite p/t nets [May84], there does not exist a *reachability-preserving* translation into finite p/t nets for any class of processes subsuming those of depth one.

We argue that any reasonable extension of mixed-bounded processes will already subsume those of depth one. Reconsider the counter machine encoding, it exploits two features that mixed-bounded processes are forbidden to combine.

First, in a process bunch $\nu a.(PB\lfloor a, \tilde{c}_x \rfloor \mid \overline{a} \mid \overline{a})$ the number of processes $\overline{a}$ under the restriction may be unbounded. Second, arbitrarily many instances of νa may be generated. If either of the conditions is dropped, the resulting process is mixed bounded. In case νa is shared by a bounded number of processes $\overline{a}$, we translate the name by the structural semantics. If finitely many instances of νa are generated, we use the concurrency semantics (cf. Examples 1 and 2). Hence, mixed-bounded processes form the borderline to finite p/t nets.

6 Discussion

Combining the structural semantics in [Mey09] and a new concurrency semantics yields a mixed semantics that finitely represents the mixed-bounded processes (Theorem 4). They generalise (Proposition 1) structurally stationary and restriction-bounded processes, the latter are finitely represented by the new concurrency semantics (Theorem 2). As it is not possible to extend mixed-bounded processes without losing reachability (Theorem 5), the class defines the borderline to finite p/t nets. Since mixed-bounded processes are bounded in depth (Proposition 2), also this class is more expressive than finite p/t nets, Figure 1.

We use a π-calculus with guarded choice and step-unwinding recursion. The former permits an elegant definition of fragments and the latter gives us decidability of structural congruence. However, both restrictions do not delimit the computational expressiveness of the π-calculus, which is the focus of the paper, but are made for technical convenience.

The definition of the mixed semantics relies on a typing mechanism for restricted names. In our tool Petruchio [Pet08], an approximate algorithm infers the types automatically. They need not be given by the user.

Finally, our implementation does not rely on the often infinite name-aware transition system, but computes the mixed semantics as a least fixed point on the set of Petri nets. Starting with the initially marked places it adds transitions and places where appropriate. A coverability graph allows us to compute the simultaneously markable places, and we currently experiment with more efficient algorithms. The compilation terminates iff the process is mixed bounded.

References

[AM02] R. Amadio and C. Meyssonnier. On decidability of the control reachability problem in the asynchronous π-calculus. *Nord. J. Comp.*, 9(1):70–101, 2002.

[BG95] N. Busi and R. Gorrieri. A Petri net semantics for π-calculus. In *Proc. of CONCUR*, volume 962 of *LNCS*, pages 145–159. Springer, 1995.

[BG09] N. Busi and R. Gorrieri. Distributed semantics for the π-calculus based on Petri nets with inhibitor arcs. *J. Log. Alg. Prog.*, 78(1):138–162, 2009.

[BGZ03] N. Busi, M. Gabbrielli, and G. Zavattaro. Replication vs. recursive definitions in channel based calculi. In *Proc. of ICALP*, volume 2719 of *LNCS*, pages 133–144. Springer, 2003.

[BGZ04] N. Busi, M. Gabbrielli, and G. Zavattaro. Comparing recursion, replication, and iteration in process calculi. In *Proc. of ICALP*, volume 3142 of *LNCS*, pages 307–319. Springer, 2004.

[Dam96] M. Dam. Model checking mobile processes. *Inf. Comp.*, 129(1):35–51, 1996.

[DFS98] C. Dufourd, A. Finkel, and Ph. Schnoebelen. Reset nets between decidability and undecidability. In *Proc. of ICALP*, volume 1443 of *LNCS*, pages 103–115. Springer, 1998.

[DKK08] R. Devillers, H. Klaudel, and M. Koutny. A compositional Petri net translation of general π-calculus terms. *For. Asp. Comp.*, 20(4–5):429–450, 2008.

[EG99] J. Engelfriet and T. Gelsema. Multisets and structural congruence of the pi-calculus with replication. *Theor. Comp. Sci.*, 211(1-2):311–337, 1999.

[EG04] J. Engelfriet and T. Gelsema. A new natural structural congruence in the pi-calculus with replication. *Acta Inf.*, 40(6):385–430, 2004.

[Eng96] J. Engelfriet. A multiset semantics for the pi-calculus with replication. *Theor. Comp. Sci.*, 153(1-2):65–94, 1996.

[FGMP03] G.-L. Ferrari, S. Gnesi, U. Montanari, and M. Pistore. A model-checking verification environment for mobile processes. *ACM Trans. Softw. Eng. Methodol.*, 12(4):440–473, 2003.

[Fok07] W. Fokkink. *Modelling Distributed Systems*. Springer, 2007.

[KKN06] V. Khomenko, M. Koutny, and A. Niaouris. Applying Petri net unfoldings for verification of mobile systems. In *Proc. of MOCA*, Bericht FBI-HH-B-267/06, pages 161–178. University of Hamburg, 2006.

[May84] E. W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM J. Comp.*, 13(3):441–460, 1984.

[Mey08a] R. Meyer. On boundedness in depth in the π-calculus. In *Proc. of IFIP TCS*, volume 273 of *IFIP*, pages 477–489. Springer, 2008.

[Mey08b] R. Meyer. *Structural Stationarity in the π-calculus.* PhD thesis, Department of Computing Science, University of Oldenburg, 2008.

[Mey09] R. Meyer. A theory of structural stationarity in the π-calculus. *Acta Inf.*, 46(2):87–137, 2009.

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

[Mil99] R. Milner. *Communicating and Mobile Systems: the π-Calculus.* CUP, 1999.

[Min67] M. Minsky. *Computation: Finite and Infinite Machines.* Prentice Hall, 1967.

[MKS09] R. Meyer, V. Khomenko, and T. Strazny. A practical approach to verification of mobile systems using net unfoldings. *Fund. Inf.*, 2009. To appear.

[MP95] U. Montanari and M. Pistore. Checking bisimilarity for finitary π-calculus. In *Proc. of CONCUR*, volume 962 of *LNCS*, pages 42–56. Springer, 1995.

[Pet08] PETRUCHIO: http://petruchio.informatik.uni-oldenburg.de, since 2008.

[Pis99] M. Pistore. *History Dependent Automata.* PhD thesis, Dipartimento di Informatica, Università di Pisa, 1999.

[SW01] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes.* CUP, 2001.