Acta Informatica (2009) 46:87–137
DOI 10.1007/s00236-009-0091-x

ORIGINAL ARTICLE

A theory of structural stationarity in the π-Calculus

Roland Meyer

Received: 2 August 2007 / Accepted: 7 January 2009 / Published online: 13 March 2009

Abstract Automata-theoretic representations have proven useful in the automatic and exact analysis of computing systems. We propose a new semantical mapping of π-Calculus processes into place/transition Petri nets. Our translation exploits the connections created by restricted names and can yield finite nets even for processes with unbounded name and unbounded process creation. The property of structural stationarity characterises the processes mapped to finite nets. We provide exact conditions for structural stationarity using novel characteristic functions. As application of the theory, we identify a rich syntactic class of structurally stationary processes, called finite handler processes. Our Petri net translation facilitates the automatic verification of a case study modelled in this class.

1 Introduction

Automated verification techniques have been successfully applied to ensure desired behaviour of various kinds of systems over the past few decades. In particular, automata-based approaches have received much attention. Our motivation is to verify systems with dynamically changing structure with the help of automata-theoretic representations.

The π-Calculus is a well-established process algebra for modelling systems with changing structure. For process algebras the investigation of automata-theoretic models has a long standing tradition. The classical question was to find system representations that reflect the concurrency of processes (cf. [35] for an overview). We investigate a translation of π-Calculus processes into place/transition Petri nets that represents the connection structure instead. This research leads to several insights.

This work was supported by the German Research Council (DFG) as part of the Graduate School "TrustSoft" (GRK 1076/1).

R. Meyer (✉)
Department of Computing Science, University of Oldenburg,
Oldenburg, Germany
e-mail: roland.meyer@informatik.uni-oldenburg.de

With the different viewpoint of structure, processes are finitely represented if and only if there are finitely many substructures every reachable process consists of. Those processes are called *structurally stationary*. In particular, this means that unbounded name creation and unbounded process creation do not necessarily imply infinite automata-theoretic representations. Our interest in finiteness is based on the observation that all automated verification methods for Petri nets rely on this constraint.

We give two alternative characterisations of structural stationarity that refer to the parallel composition and the restriction operator, respectively. The first one eases the proof of structural stationarity. The second one reveals that infinity of the semantics has two sources: unbounded *breadth* and unbounded *depth*. Unbounded breadth is caused by unbounded distribution of restricted names. Unbounded depth is caused by unboundedly long chains of processes connected by restricted names.

The systems we aim to verify can be found in the traffic control domain. They naturally have a client-server architecture. We design the syntactic class of *finite handler processes* to model these systems. An application of the first characterisation shows that finite handler processes are structurally stationary.

On a simplified model of a highway control system, we evaluate the usefulness of our novel translation for the automatic verification. We infer non-trivial properties of various kinds (occurrence number, topological, and temporal properties) automatically considering the system under our semantics. To establish the properties, efficient verification algorithms that inspect the graph structure of the net are sufficient.

To sum up, our contributions are as follows.

- We define a semantical mapping of π-Calculus processes into place/transition Petri nets. Our semantics reflects the connection structure of processes. The transition system of a process and that of its Petri net semantics are isomorphic.
- On a case study we demonstrate the application of the semantics to automatically prove different classes of system properties. We show that efficient algorithms can be exploited in the verification.
- The semantics is finite if and only if the process under consideration has the semantical property of structural stationarity. We give two complete characterisations of structural stationarity.
- We design a rich syntactic class, called finite handler processes, that we prove to be structurally stationary. Our case study is modelled in this class.

The paper is organised as follows. After recalling the π-Calculus and Petri nets in Sect. 2, we propose our novel semantics in Sect. 3. Section 4 applies the translation to automatically prove properties of a case study. Structural stationarity characterises the processes that are finitely represented under the semantics. We investigate this property in Sect. 5. The class of finite handler processes is considered in Sect. 6. Section 7 discusses related work, and Sect. 8 concludes the paper. Proofs and details missing in the paper can be found in the appendix.

1.1 Case study

For illustrating the theory developed in this paper, we use a simplified model of a highway control system [23]. The scenario is as follows. A vehicle enters a highway and communicates its goal location to a central device. The device responds by sending a path the vehicle has to follow. To do so, it chooses one of three elementary manoeuvres: change the current lane, merge with preceding vehicles into so-called platoons, and split platoons. Thanks to

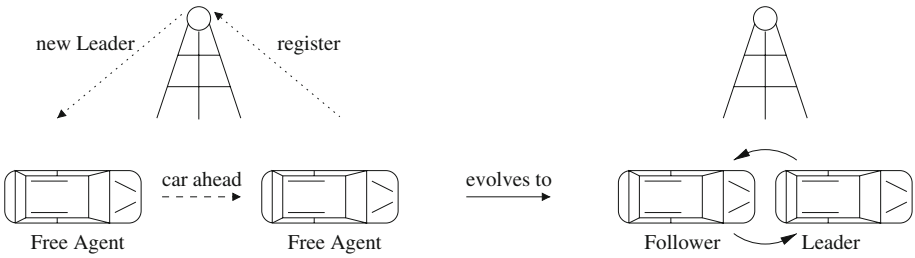

Fig. 1 Illustration of the merge manoeuvre

computer-supported control, cars in platoons drive close to each other. This gives hope for better utilisation of highways and shorter travelling times when the system is installed.

We focus on the merge manoeuvre as illustrated in Fig. 1. For its description we call single vehicles *free agents*. When a free agent detects another one driving in front, depicted by a *car ahead* message, it contacts that vehicle with the request to merge. If the leading vehicle agrees, the now called *follower* speeds up to drive closely behind the *leader*. They maintain a permanent connection, indicated by the two arrows between follower and leader in Fig. 1. The two cars form a *car platoon*. In our model, we only merge free agents whereas the original case study also merges platoons up to a given size. The theory developed in this paper allows us to verify those systems with known bounds as well. We simplified the case study for the sake of brevity and clarity.

The verification in [23] relies on proper connections between the vehicles. Therefore, the results achieved are only valid modulo this assumption. We present a model where the connections are built up appropriately with the help of central control units managing sections of the highway. These units are represented by the masts in Fig. 1. Free agents entering a section register at the unit and receive the channel for contacting a preceding vehicle from the unit. To limit the number of messages in our case study, we let the central control unit send the car ahead message with the new leader as a parameter instead of sending two messages.

2 Preliminaries

Since we translate π-Calculus processes into place/transition Petri nets, we recall both of them in this section.

2.1 π-Calculus

We use a monadic π-Calculus variant with parameterised recursion as proposed in [40]. Given an infinite set of names $(a, b, x, y, \ldots \in) \mathcal{N}$, the *prefixes* π of the π-Calculus are defined by

$$\pi ::= \overline{x}\langle y \rangle \mid x(y) \mid \tau.$$

The output action prefix $\overline{x}\langle y \rangle$ sends name y along channel x, the input action prefix $x(y)$ receives a name via x that replaces y, and prefix τ performs an internal action.

Let $\tilde{a}$ abbreviate a finite sequence of names, $\tilde{a} = a_1, \ldots, a_n$, which is also treated as a set where required, $\tilde{a} = \bigcup_{i=1}^{n} \{a_i\}$. To define parameterised recursion, we use upper-case letters K as *process identifiers*. A process identifier represents a process P via a recursive definition $K(\tilde{x}) := P$, where the elements in $\tilde{x}$ are distinct. The term $K\lfloor \tilde{a} \rfloor$ is a *call* to the

process identifier, which results in the process P where the names $\tilde{x}$ are replaced by $\tilde{a}$. The remaining operators are standard. In a *parallel composition* $P_1 \mid P_2$, the processes P_1 and P_2 communicate via pairs of send and receive prefixes. The process offering a *choice* between the prefixes π_i that can be used for communication is $\Sigma_{i\in I}\pi_i.P_i$. After a prefix π_i has been consumed, the process behaves like P_i. The *restriction* operator $\nu a.P$ converts the name a in P into a private name. It is different from all names in other processes.

Definition 1 (The π-Calculus) The set of *π-Calculus processes* $(P, Q \in)\mathcal{P}$ is defined by the grammar

$$P ::= \Sigma_{i\in I}\pi_i.P_i \;\mid\; P_1 \mid P_2 \;\mid\; \nu a.P \;\mid\; K\lfloor\tilde{a}\rfloor,$$

where $I \subseteq \mathbb{N}$ is a finite index set. ◇

We abbreviate arbitrary sums with M or N, the empty sum ($I = \emptyset$) with $\mathbf{0}$. We omit any pending $\mathbf{0}$, i.e., we write π instead of $\pi.\mathbf{0}$. The choice composition involving a distinguished term $\pi.P$ is denoted by $\pi.P + M$. By $\Pi^k P$ we denote the parallel composition of $k \geq 1$ processes P, by $\Pi_{i=1}^n P_i$ the parallel composition $P_1 \mid \ldots \mid P_n$ with $n \geq 1$, and by $\Pi_{i\in I} P_i$ we denote $P_{i_1} \mid \ldots \mid P_{i_n}$, where $I = \{i_1, \ldots, i_n\} \subseteq \mathbb{N}$ with $i_1 < \ldots < i_n$. As a special case, $\Pi_{i\in\emptyset} P_i := \mathbf{0}$. A sequence of restrictions $\nu a_1 \ldots \nu a_n.P$ is abbreviated by $\nu\tilde{a}.P$ where $\tilde{a} = a_1, \ldots, a_n$. To avoid brackets, we define that (1) prefix π binds stronger than choice composition $+$ and (2) choice composition as well as restriction νa bind stronger than parallel composition $\mid$.

Our definition of choice composition uses only prefixed processes $\pi.P$ as alternatives. In the literature, this well-accepted restriction is called guarded choice. It ensures that before process P_i can be executed in $\pi_1.P_1 + \cdots + \pi_n.P_n$ prefix π_i has to be consumed. The definition excludes choices of the form $(P1 \mid P2) + P3$, for which it is known to be hard to define suitable Petri net semantics [35] and which are considered of minor practical importance [40].

The input action $x(y)$ and the restriction νa *bind* the names y and a, respectively. There is at most one restriction or input prefix binding a name. All names that are not bound by an input action or a restriction are *free*. The *set of free names* of a process P is denoted by $fn(P)$, the *set of bound names* by $bn(P)$. Without loss of generality, we assume the free names disjoint from the set of bound names. We call a process P *closed* if $fn(P) = \emptyset$. The *set of all closed π-Calculus processes* is $\mathcal{CP}$. For the process identifier $K(\tilde{x}) := P$ we demand $fn(P) \subseteq \tilde{x}$. By convention, a process depends on finitely many recursive definitions.

Substitutions are functions that rename free names in a process. Since we need substitutions in Sect. 5, we recall their definition following [40]. A substitution σ is a mapping from names to names, i.e., $\sigma : \mathcal{N} \to \mathcal{N}$. Let $x\sigma$ denote the image of x under σ. If we give a domain and a range, $\sigma : A \to B$ where $A, B \subseteq \mathcal{N}$, we demand $x\sigma \in B$ if $x \in A$ and $x\sigma = x$ otherwise. An explicitly defined substitution $\sigma = \{a_1, \ldots, a_n/x_1, \ldots, x_n\}$ maps x_i to a_i, i.e., $\sigma : \{x_1, \ldots, x_n\} \to \{a_1, \ldots, a_n\}$ with $x_i\sigma = a_i$. If we apply a substitution $\sigma : A \to B$ to a process P, we demand the set of names in σ to be disjoint from the bound names in P, $(A \cup B) \cap bn(P) = \emptyset$.

Definition 2 (Application of substitutions) Given a substitution $\sigma : \mathcal{N} \to \mathcal{N}$ and a process P, we define $P\sigma$ inductively by:

$$\begin{aligned} \tau\sigma &:= \tau & x(y)\sigma &:= x\sigma(y) \\ \overline{x}\langle y\rangle\sigma &:= \overline{x\sigma}\langle y\sigma\rangle & (\Sigma_{i\in I}\pi_i.P_i)\sigma &:= \Sigma_{i\in I}\pi_i\sigma.P_i\sigma \\ (P \mid Q)\sigma &:= P\sigma \mid Q\sigma & (\nu a.P)\sigma &:= \nu a.(P\sigma). \end{aligned}$$ ◇

The application of a substitution σ to a set of processes $\{P_1, P_2, \ldots\}$ is defined elementwise, i.e., $\{P_1, P_2, \ldots\}\sigma := \{P_1\sigma, P_2\sigma, \ldots\}$.

As usual, we employ the structural congruence relation for the definition of the operational behaviour of π-Calculus processes. For the implementation of our semantical mapping it is important to note that structural congruence is decidable in our setting [27].

Definition 3 (Structural Congruence) *Structural congruence*, denoted by $\equiv$, is the least congruence relation on processes where α-conversion of bound names is allowed, $+$ and $\mid$ are commutative and associative with $\mathbf{0}$ as neutral element, and the following laws for restriction hold:

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

The latter law is called *scope extrusion.*

Note that structural congruence allows for removing restricted names that do not occur free in a process, i.e., $\nu a.P \equiv P$ if $a \notin fn(P)$. In Sect. 3.2, we exploit this congruence for computing the restricted form of a process.

Of particular interest in this paper are *sequential processes*, i.e., non-empty sums $\Sigma_{i\in I\neq\emptyset}\pi_i.P_i$ and calls to process identifiers $K\lfloor\tilde{a}\rfloor$. The process $K\lfloor\tilde{a}\rfloor$ is called sequential, even if K is defined by a parallel composition, $K(\tilde{x}) := P \mid Q$. The intuition is that a call to an identifier yields a reaction in the reaction relation defined below, which means recursion is guarded by a τ action. We use the function $\mathcal{S} : \mathcal{P} \to \mathbb{P}(\mathcal{P})$ to refer to the sequential processes inside a given process.

Definition 4 ($\mathcal{S} : \mathcal{P} \to \mathbb{P}(\mathcal{P})$) The set of *sequential processes in a process* $P \in \mathcal{P}$ is given by $\mathcal{S}(P)$ defined as follows:

$$\mathcal{S}(\mathbf{0}) := \emptyset \qquad \mathcal{S}(P_1 \mid P_2) := \mathcal{S}(P_1) \cup \mathcal{S}(P_2)$$
$$\mathcal{S}(K\lfloor\tilde{a}\rfloor) := \{K\lfloor\tilde{a}\rfloor\} \qquad \mathcal{S}(\nu a.P) := \mathcal{S}(P)$$
$$\mathcal{S}(\Sigma_{i\in I\neq\emptyset}\pi_i.P_i) := \{\Sigma_{i\in I\neq\emptyset}\pi_i.P_i\}. \qquad \diamond$$

The application of substitutions is compatible with the computation of the sequential processes.

Lemma 1 *Given $P \in \mathcal{P}$ and $\sigma : fn(P) \to \mathcal{N}$. Then $\mathcal{S}(P\sigma) = \mathcal{S}(P)\sigma$.*

The following Lemma 2 shows that every process P can be rewritten as a parallel composition of its sequential processes. The idea is to compute the *standard form* [30], where restricted names are dragged over all parallel compositions. Since we assume that a name is bound at most once and that the bound names are disjoint with the free names, α-conversion is not required.

Lemma 2 *Every process $P \in \mathcal{P}$ is structurally congruent with a process $\nu\tilde{a}.(\Pi_{i\in I} P_i)$ where $\tilde{a} \subseteq bn(P)$ and for each i it holds $P_i \in \mathcal{S}(P)$.*

The behaviour of π-Calculus processes is determined by the reaction relation $\to$. Rule (Struct) closes $\to$ under structural congruence.

$$
\begin{aligned}
CREATE(CFA) &:= FA\lfloor CFA\rfloor \mid CREATE\lfloor CFA\rfloor \\
MERGE(CFA) &:= CFA(id_x).id_x(ca_x).id_x(rq_x). \\
&\qquad CFA(id_y).id_y(ca_y).id_y(rq_y).\overline{ca_y}\langle rq_x\rangle.MERGE\lfloor CFA\rfloor \\
FA(CFA) &:= \nu id, ca, rq.\overline{CFA}\langle id\rangle.\overline{id}\langle ca\rangle.\overline{id}\langle rq\rangle. \\
&\qquad (ca(rqnl).REQ\lfloor id, rqnl\rfloor + rq(nf).\overline{nf}\langle id\rangle.LD\lfloor id, nf\rfloor) \\
REQ(id, rqnl) &:= \overline{rqnl}\langle id\rangle.id(nl).FL\lfloor id, nl\rfloor.
\end{aligned}
$$

Fig. 2 The car platoon system modelled in the π-Calculus

Definition 5 (Reaction relation) The *reaction relation* $\rightarrow \subseteq \mathcal{P}\times\mathcal{P}$ is defined by the following rules:

$$\text{(Tau)}\ \tau.P + M \rightarrow P$$

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

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

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

$$\text{(Struct)}\ \frac{P \rightarrow P'}{Q \rightarrow Q'}, \text{ if } P \equiv Q \text{ and } P' \equiv Q'. \qquad \diamond$$

The relation $\rightarrow$ is image-finite, i.e., for every process P there are up to structural congruence only finitely many processes Q with $P \rightarrow Q$. When we refer to the *transition system of a process* $P \in \mathcal{P}$, we mean the transition system consisting of all processes reachable from P with the reaction relation, factorised by structural congruence. We denote the *set of all processes reachable from P with the reaction relation* by $Reach(P) := \{Q \mid P \rightarrow^* Q\}$ where $\rightarrow^* \subseteq \mathcal{P}\times\mathcal{P}$ is the reflexive and transitive closure of $\rightarrow$.

Example 1 (Car platoon) We consider the example of two free agents merging into a platoon. This case study is specified by the π-Calculus process $CREATE\lfloor CFA\rfloor \mid MERGE\lfloor CFA\rfloor$ that uses the defining equations given in Fig. 2. The *CREATE* process recursively generates new free agents. These free agents have an *id*, a channel *ca*, and a channel *rq*. The channel *ca* models a car ahead message of the environment. The request channel *rq* is used by a second free agent to issue a request to merge into a platoon. Free agents register at a *MERGE* process. They pass their *id* to establish a private connection and then send the *ca* and *rq* channels. At some point, the *MERGE* process sends a *ca* message to the second free agent that registered. It contains the *rq* channel of the first free agent. The agent receives this message and becomes a request process *REQ* that contacts the other free agent on the *rq* channel. If this agent accepts the request to merge, the *REQ* process turns into a follower *FL* and the first car into a leader *LD*. This finishes the merge manoeuvre. We omit the defining equations for *FL* and *LD* as they are unimportant for the merge procedure. $\diamond$

2.2 Place/transition Petri nets

Petri nets are an automata-theoretic model for distributed systems. They generalise finite automata with the idea of distributed states and explicit synchronisation of transitions. We use standard notions of place/transition Petri nets with weights on arcs (cf. [37]).

Definition 6 (Place/transition Petri net) An *unmarked place/transition Petri net* is a triple (S, T, λ), where

S is a (potentially infinite) set of *places*,
T is a (potentially infinite) set of *transitions* disjoint from S,
$\lambda : (S \times T) \cup (T \times S) \to \mathbb{N}$ is a total *weight function* giving the number of arcs from $s \in S$ to $t \in T$ and vice versa.

The states of Petri nets, called *markings*, are mappings $(M \in) \mathbb{N}^S$. The *support* of a marking is $supp(M) := \{s \in S \mid M(s) > 0\}$. We refer to (unmarked) place/transition Petri nets with an initial marking M_0 as *marked* place/transition Petri nets or Petri nets. We denote the *set of all Petri nets* by PN. We call a Petri net *finite* if S and T are finite. $\diamond$

The graphical representation of a Petri net is defined as follows. Every place $s \in S$ is represented by a circle, every transition $t \in T$ by a box. If $\lambda(s, t) = 0$, no arc from s to t is drawn. We draw an unlabelled arc if $\lambda(s, t) = 1$, and an arc labelled by k if $\lambda(s, t) = k > 1$. For a given marking M with $M(s) = k$ we say that place s carries k *tokens*. We represent tokens by black dots inside the circles for the places. Figure 5 shows a Petri net. In Sect. 3, we develop the correspondence of this net with the process in Fig. 2.

To define the behaviour of a Petri net, we need the set of places a transition consumes tokens from. This *preset* of a transition $t \in T$ is ${}^\bullet t := \{s \in S \mid \lambda(s, t) > 0\}$. The places a transition t produces tokens on are in the *postset* of t, $t^\bullet := \{s \in S \mid \lambda(t, s) > 0\}$. Pre- and postsets of places are defined similarly. The behaviour of a Petri net is given by a transition relation. It states how marking M changes to M' when executing transition t.

Definition 7 (Transition relation) Consider the Petri net (S, T, λ, M_0). A transition $t \in T$ is *enabled* under a marking M, if $M(s) \geq \lambda(s, t)$ for all $s \in {}^\bullet t$. The *transition relation* $[\rangle \subseteq \mathbb{N}^S \times T \times \mathbb{N}^S$ is defined by:

$$M[t\rangle M' \text{ iff } t \text{ is enabled under } M \text{ and}$$
$$M'(s) = M(s) - \lambda(s, t) + \lambda(t, s), \text{ for all } s \in S. \qquad \diamond$$

The *transition system* of a Petri net (S, T, λ, M_0) consists of all markings reachable from M_0 with the transition relation. The identity of transitions is omitted, i.e., $M \to M'$ iff $M[t\rangle M'$ for some $t \in T$. We denote the *set of all reachable markings* by $Reach((S, T, \lambda, M_0)) := \{M \mid M_0 \to^* M\}$, where $\to^* \subseteq \mathbb{N}^S \times \mathbb{N}^S$ is the reflexive and transitive closure of $\to$.

For the verification of finite nets, the theory of S-Invariants is available. It is standard in Petri net theory and can be found in introductory books (e.g. [37]).

Definition 8 (S-Invariant) For a Petri net (S, T, λ, M_0), the *incidence matrix* is $C : S \times T \to \mathbb{Z}$ with $C(s, t) := \lambda(t, s) - \lambda(s, t)$. An *S-Invariant* of the Petri net is a vector $x \in \mathbb{Z}^S$ with $C^t \cdot x = 0$, where C^t is the transposed incidence matrix and $\_ \cdot \_$ is the matrix product. $\diamond$

An S-invariant gives a weight for the tokens on each place. The fundamental property of S-Invariants is that the weighted sum of tokens is invariant under the transition relation.

Lemma 3 (Fundamental property of S-Invariants [37]) *Let M, M' be two markings of a Petri net with $M \to^* M'$, let I be an S-Invariant. Then $I^t \cdot M = I^t \cdot M'$ holds.*

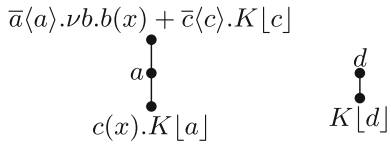

Fig. 3 The structure $\mathcal{G}[\![P]\!]$ of process P in Example 2

3 A structural semantics for the π-Calculus

To begin with, we recall the structure of π-Calculus processes along the lines of [30,40] and informally explain the idea of our semantics. Then we give it a rigorous formal definition based on a normal form for processes.

3.1 The structure of π-Calculus processes

The *structure of a π-Calculus process* P is a bipartite graph $\mathcal{G}[\![P]\!]$ which makes the use of restricted names in sequential processes explicit. It is obtained as follows. We draw a node labelled by Q for every sequential process $Q = \Sigma_{i\in I\neq\emptyset}\pi_i.P_i$ or $Q = K\lfloor\tilde{a}\rfloor$ in $\mathcal{S}(P)$ and add a node labelled by a for every restricted name νa outside choice compositions $\Sigma_{i\in I\neq\emptyset}\pi_i.P_i$. An arc is inserted between a node Q and a node a, if the name is free in the process, $a \in fn(Q)$. Due to name passing, process creation, and process destruction, this graph structure may change during system execution.

Example 2 (Structure of a process) We illustrate this graph representation on an example. The process

$$P = \nu a.(\overline{a}\langle a\rangle.\nu b.b(x) + \overline{c}\langle c\rangle.K\lfloor c\rfloor \mid c(x).K\lfloor a\rfloor \mid \nu d.K\lfloor d\rfloor)$$

contains the three sequential processes $\overline{a}\langle a\rangle.\nu b.b(x)+\overline{c}\langle c\rangle.K\lfloor c\rfloor$, $c(x).K\lfloor a\rfloor$, and $K\lfloor d\rfloor$. The restricted names νa and νd are outside choice compositions while νb is inside $\overline{a}\langle a\rangle.\nu b.b(x)+\overline{c}\langle c\rangle.K\lfloor c\rfloor$. This explains the nodes in the graph $\mathcal{G}[\![P]\!]$ in Fig. 3. The name a is free in $\overline{a}\langle a\rangle.\nu b.b(x) + \overline{c}\langle c\rangle.K\lfloor c\rfloor$ and in $c(x).K\lfloor a\rfloor$, thus the corresponding nodes are connected with the node a. Similarly, the nodes for $K\lfloor d\rfloor$ and d are connected. ◇

As the example shows, several unconnected subgraphs represent one process. This means, mobile systems are modelled by independent parts communicating over public channels only. The idea of our semantics is to represent each such subgraph by a place in a Petri net. We then obtain the overall process by putting tokens on the places of the subgraphs, one for each occurrence of the graph structure in the overall process.

We use graphs as intuition for our semantics. Formally, we show that every process can be assumed to be in *restricted form*. Normal forms similar to ours are defined in [13,15], but with different aims. We discuss the relationship to the work of Engelfriet and Gelsema in Sect. 7.

3.2 The restricted form of processes

With the restricted form, we capture the notion of unconnected subgraphs introduced in the previous Sect. 3.1. It serves us for the definition of our semantics and also permits the definition of the characteristic functions depth and breadth in Sect. 5.2. The idea of the restricted form is to minimise the scopes of restricted names outside choice compositions. This results in a process, where the topmost parallel components are the unconnected subgraphs. We call them *fragments*.

Example 3 (Restricted form) Consider the process in Example 2:

$$\begin{aligned} & \nu a.(\overline{a}\langle a\rangle.\nu b.b(x) + \overline{c}\langle c\rangle.K\lfloor c\rfloor \mid c(x).K\lfloor a\rfloor \mid \nu d.K\lfloor d\rfloor) \\ \equiv\ & \nu a.(\overline{a}\langle a\rangle.\nu b.b(x) + \overline{c}\langle c\rangle.K\lfloor c\rfloor \mid c(x).K\lfloor a\rfloor) \mid \nu d.K\lfloor d\rfloor. \end{aligned}$$

The latter process is in restricted form as the scopes of νa and νd are minimal. Fragment $\nu a.(\overline{a}\langle a\rangle.\nu b.b(x)+\overline{c}\langle c\rangle.K\lfloor c\rfloor \mid c(x).K\lfloor a\rfloor)$ corresponds to the subgraph to the left in Fig. 3, fragment $\nu d.K\lfloor d\rfloor$ is the subgraph depicted to the right. ◇

Formally, sequential processes, i.e., choice compositions $\Sigma_{i\in I\neq\emptyset}\pi_i.P_i$ and calls to process identifiers $K\lfloor\tilde{a}\rfloor$, are fragments. On the graph level they are nodes. Restricting a name in a parallel composition of fragments, $\nu a.(F_1 \mid \ldots \mid F_n)$, is possible only if the name is free in all fragments, $a \in fn(F_i)$ for all i. This ensures the graphs of all fragments F_i are connected with the node for the restricted name νa. So the graph of $\nu a.(F_1 \mid \ldots \mid F_n)$ is connected. A process P_ν in restricted form is a parallel composition of fragments, $P_\nu = \Pi_{i\in I} F_i$. Since different fragments do not share restricted names, P_ν is represented by the graphs of the fragments F_i, which are not connected.

Definition 9 (Restricted form) The set of *fragments* $\mathcal{P}_\mathcal{F}$ with typical elements F, G, H is defined inductively by:

$$F ::= \Sigma_{i\in I\neq\emptyset}\pi_i.P_i \ \mid\ K\lfloor\tilde{a}\rfloor \ \mid\ \nu a.(F_1 \mid \ldots \mid F_n),$$

where $a \in fn(F_i)$ for all $1 \leq i \leq n$. A process $P_\nu = \Pi_{i\in I} F_i$ is in *restricted form*. We refer to the fragments in P_ν by $Frag(P_\nu) := \cup_{i\in I}\{F_i\}$. The *set of all processes in restricted form* is $(\mathcal{P}_\mathcal{F} \subseteq)\mathcal{P}_\nu$. ◇

To transform a given process into a process in restricted form via structural congruence, we employ the recursive function $\nu : \mathcal{P} \to \mathcal{P}_\nu$. It uses the rule for scope extrusion to shrink the scopes of restrictions and—to avoid cumbersome case distinctions—tacitly removes parallel compositions of $\mathbf{0}$:

$$\begin{aligned} \nu(\Sigma_{i\in I}\pi_i.P_i) &:= \Sigma_{i\in I}\pi_i.P_i & \qquad \nu(P_1 \mid P_2) &:= \nu(P_1) \mid \nu(P_2) \\ \nu(K\lfloor\tilde{a}\rfloor) &:= K\lfloor\tilde{a}\rfloor & \qquad \nu(\nu a.P) &:= \nu a.\left(\Pi_{i\in I_a} F_i\right) \mid \Pi_{i\in I\setminus I_a} F_i, \end{aligned}$$

where $\nu(P) = \Pi_{i\in I} F_i$ and the set $I_a \subseteq I$ contains the indices of those fragments that have a as a free name, i.e., $i \in I_a$ if and only if $a \in fn(F_i)$. That the function removes parallel compositions of $\mathbf{0}$ means $\nu(P_1 \mid P_2)$ is defined by $\nu(P_1)$ instead of $\nu(P_1) \mid \mathbf{0}$, if $\nu(P_2) = \mathbf{0}$. Similarly, $\nu(\nu a.P) := \Pi_{i\in I} F_i$ if $I_a = \emptyset$. The following lemma states that $\nu(P)$ is in fact in restricted form and structurally congruent with P. Furthermore, the function does not change processes in restricted form, i.e., $\nu(P_\nu) = P_\nu$.

Lemma 4 *For a process $P \in \mathcal{P}$ we have $\nu(P) \in \mathcal{P}_\nu$ and $\nu(P) \equiv P$. For $P_\nu \in \mathcal{P}_\nu$ even $\nu(P_\nu) = P_\nu$ holds.*

The restricted form is only invariant under structural congruence up to reordering and rewriting of fragments, i.e., $P \equiv Q$ does not imply $\nu(P) = \nu(Q)$ but it implies $\nu(P) \mathrel{\widehat{\equiv}} \nu(Q)$, where we define the relation $\mathrel{\widehat{\equiv}}$ as follows.

Definition 10 (Restricted equivalence) The *restricted equivalence relation* $\mathrel{\widehat{\equiv}} \subseteq \mathcal{P}_\nu \times \mathcal{P}_\nu$ is the smallest equivalence on processes in restricted form that satisfies commutativity and associativity of parallel compositions, $P_\nu \mid Q_\nu \mathrel{\widehat{\equiv}} Q_\nu \mid P_\nu$ and $(P_\nu \mid Q_\nu) \mid R_\nu \mathrel{\widehat{\equiv}} P_\nu \mid (Q_\nu \mid R_\nu)$,

and permits replacing fragments by structurally congruent ones, $F \mid P_\nu \mathrel{\widehat{\equiv}} G \mid P_\nu$, if $F \equiv G$. ◇

We illustrate the relationship between $\nu(P)$ and $\nu(Q)$ with $P \equiv Q$ on an example.

Example 4 (Invariance of $\nu(-)$ under $\equiv$ up to $\mathrel{\widehat{\equiv}}$) Consider the process P in Example 2 and a structurally congruent process Q:

$$\begin{aligned} P &= \nu a.(\overline{a}\langle a\rangle.\nu b.b(x) + \overline{c}\langle c\rangle.K\lfloor c\rfloor \mid c(x).K\lfloor a\rfloor \mid \nu d.K\lfloor d\rfloor) \\ &\equiv \nu a.(c(x).K\lfloor a\rfloor \mid \overline{a}\langle a\rangle.\nu b.b(x) + \overline{c}\langle c\rangle.K\lfloor c\rfloor \mid \nu d.K\lfloor d\rfloor) = Q. \end{aligned}$$

The restricted form of P, $\nu(P)$, is the process in Example 3. We compare it with the restricted form of Q:

$$\begin{aligned} \nu(P) &= \nu a.(\overline{a}\langle a\rangle.\nu b.b(x) + \overline{c}\langle c\rangle.K\lfloor c\rfloor \mid c(x).K\lfloor a\rfloor) \mid \nu d.K\lfloor d\rfloor \\ &\mathrel{\widehat{\equiv}} \nu a.(c(x).K\lfloor a\rfloor \mid \overline{a}\langle a\rangle.\nu b.b(x) + \overline{c}\langle c\rangle.K\lfloor c\rfloor) \mid \nu d.K\lfloor d\rfloor = \nu(Q)\,. \end{aligned}$$

Since the first two fragments, $\nu a.(\overline{a}\langle a\rangle.\nu b.b(x) + \overline{c}\langle c\rangle.K\lfloor c\rfloor \mid c(x).K\lfloor a\rfloor)$ and $\nu a.(c(x).K\lfloor a\rfloor \mid \overline{a}\langle a\rangle.\nu b.b(x) + \overline{c}\langle c\rangle.K\lfloor c\rfloor)$, are not (syntactically) equal but structurally congruent, $\nu(P) \neq \nu(Q)$ but $\nu(P) \mathrel{\widehat{\equiv}} \nu(Q)$ holds. ◇

Proposition 1 states the discussed invariance of the restricted form up to $\mathrel{\widehat{\equiv}}$, $P \equiv Q$ implies $\nu(P) \mathrel{\widehat{\equiv}} \nu(Q)$. In fact, the restricted equivalence relation $\mathrel{\widehat{\equiv}}$ characterises structural congruence, i.e., also $\nu(P) \mathrel{\widehat{\equiv}} \nu(Q)$ implies $P \equiv Q$.

Proposition 1 *For $P, Q \in \mathcal{P}$ holds: $P \equiv Q$ if and only if $\nu(P) \mathrel{\widehat{\equiv}} \nu(Q)$.*

3.3 The structural semantics

In this section, we define a Petri net semantics for the π-Calculus, i.e., a mapping $\mathcal{N} : \mathcal{P} \to PN$ that assigns to every process P a Petri net $\mathcal{N}[\![P]\!]$. The places of the net are the fragments of all processes reachable from P by the reaction relation. More precisely, we deal with classes of fragments under structural congruence.

We use two disjoint sets of transitions. Transitions of the first kind are pairs $([F'], [Q])$ of places $[F']$ and processes $[Q]$, with the condition that F' reacts to Q. These transitions represent reactions inside fragments. The second set of transitions contains pairs $([F_1 \mid F_2], [Q])$ where $[F_1]$ and $[F_2]$ are places and $F_1 \mid F_2$ reacts to Q. These transitions represent communications between fragments using public channels.

There is an arc from place $[F]$ to transition $([F'], [Q])$ if F is structurally congruent with F', i.e., $[F] = [F']$. If F is structurally congruent with F_1 and F_2, there is an arc weighted two from place $[F]$ to transition $([F_1 \mid F_2], [Q])$. In this case, fragment F_1 communicates with the structurally congruent fragment F_2 on a public channel. If F is structurally congruent with F_1 or F_2, there is an arc weighted one from place $[F]$ to transition $([F_1 \mid F_2], [Q])$. There is no arc, if F is neither structurally congruent with F_1 nor with F_2.

The number of arcs from $([F'], [Q])$ to place $[F]$ (or from $([F_1 \mid F_2], [Q])$ to place $[F]$, respectively) is determined by the number of occurrences of F in the decomposition of Q. Similarly, the initial marking of the net is determined by the decomposition of the initial process P. To formalise the notion of occurrences in a decomposition, we define the function $dec\,(P_\nu)$. It maps a congruence class of fragments $[F]$ to the number of fragments in P_ν that are in this class, e.g., if $P_\nu = F \mid G \mid F$ with $F \not\equiv G$ then $(dec\,(P_\nu))\,([F]) = 2$, $(dec\,(P_\nu))\,([G]) = 1$, and $(dec\,(P_\nu))\,([H]) = 0$ for $F \not\equiv H \not\equiv G$.

Definition 11 ($dec : \mathcal{P}_\nu \to \mathbb{N}^{\mathcal{P}_\mathcal{F}/\equiv}$) Let $P_\nu = \Pi_{i\in I} F_i$. We assign to P_ν the function $dec(P_\nu) : \mathcal{P}_\mathcal{F}/_\equiv \to \mathbb{N}$ via $(dec(P_\nu))([F]) := |I_F|$, where $I_F \subseteq I$ so that $i \in I_F$ iff $F \equiv F_i$. ◇

That $dec(-)$ is invariant under $\widehat{\equiv}$ ensures our semantics is well-defined. That it characterises $\widehat{\equiv}$ is exploited in the proof of Theorem 1.

Lemma 5 *Let $P_\nu, Q_\nu \in \mathcal{P}_\nu$. Then $P_\nu \widehat{\equiv} Q_\nu$ if and only if $dec(P_\nu) = dec(Q_\nu)$.*

By definition, $dec(-)$ is compatible with parallel composition.

Lemma 6 *Let $P_\nu, Q_\nu \in \mathcal{P}_\nu$. Then $dec(P_\nu \mid Q_\nu) = dec(P_\nu) + dec(Q_\nu)$ holds, where $(dec(P_\nu) + dec(Q_\nu))([F]) := (dec(P_\nu))([F]) + (dec(Q_\nu))([F])$.*

We are now prepared to define our structural semantics.

Definition 12 (Structural semantics $\mathcal{N} : \mathcal{P} \to PN$) The *structural semantics* is a mapping that yields a Petri net $\mathcal{N}[\![P]\!] := (S, T, \lambda, M_0)$ for every process $P \in \mathcal{P}$ as follows:

$$
\begin{aligned}
S &:= Frag(\nu(Reach(P)))/_\equiv \\
T &:= \{([F], [Q]) \in S \times \mathcal{P}/_\equiv \mid F \to Q\} \\
&\quad \cup \{([F_1 \mid F_2], [Q]) \in \mathcal{P}/_\equiv \times \mathcal{P}/_\equiv \mid [F_1], [F_2] \in S \text{ and } F_1 \mid F_2 \to Q\}.
\end{aligned}
$$

To define the weight function λ, we consider the place $[F] \in S$ and two transitions $([F'], [Q]), ([F_1 \mid F_2], [Q]) \in T$:

$$
\begin{aligned}
\lambda([F], ([F'], [Q])) &:= \left(dec\left(F'\right)\right)([F]) \\
\lambda([F], ([F_1|F_2], [Q])) &:= (dec(F_1 \mid F_2))([F]) \\
\lambda(([F'], [Q]), [F]) &:= \lambda(([F_1 \mid F_2], [Q]), [F]) := (dec(\nu(Q)))([F]) \\
M_0 &:= dec(\nu(P)).
\end{aligned}
$$
◇

We briefly discuss the definition. The weight $\lambda([F], ([F'], [Q]))$ is defined by $\left(dec\left(F'\right)\right)([F])$, which is 1 if $[F] = [F']$ and 0 otherwise. We prefer the technical definition to a case distinction because it is more convenient in the proofs (cf. proof of Lemma 9). Similarly, $\lambda([F], ([F_1 \mid F_2], [Q]))$ is defined by $(dec(F_1 \mid F_2))([F])$. In this case, $(dec(F_1 \mid F_2))([F]) = 2$ if $[F_1] = [F] = [F_2]$, $(dec(F_1 \mid F_2))([F]) = 0$ if $[F_1] \neq [F] \neq [F_2]$, and 1 otherwise. Since fragments are different from $\mathbf{0}$, a process $P \equiv \mathbf{0}$ is mapped to the net without places and transitions.

Consider a fragment F_1 that reacts to Q. This reaction is modelled by a transition $([F_1], [Q])$. But $F_1 \mid F_2$ also reacts to $Q \mid F_2$ for every fragment F_2. Thus, we additionally get a transition $([F_1 \mid F_2], [Q \mid F_2])$ for every reachable fragment $[F_2]$. Since only a loop reproducing a token on place $[F_2]$ differentiates $([F_1 \mid F_2], [Q \mid F_2])$ from $([F_1], [Q])$, the additional transitions do not change the transition system. We do not compute them. We do not exclude them by a side condition as this complicates the proof of Theorem 1.

Example 5 (Structural semantics) We illustrate our translation on a small example. Consider the process

$$
\begin{aligned}
P = {} & a(x).x(y).y(z).\overline{a}\langle c\rangle + \overline{a}\langle b\rangle \mid a(x).x(y).y(z).\overline{a}\langle c\rangle + \overline{a}\langle b\rangle \mid \\
& \nu h.\overline{b}\langle h\rangle.\overline{h}\langle b\rangle.(c(x) \mid c(x)).
\end{aligned}
$$

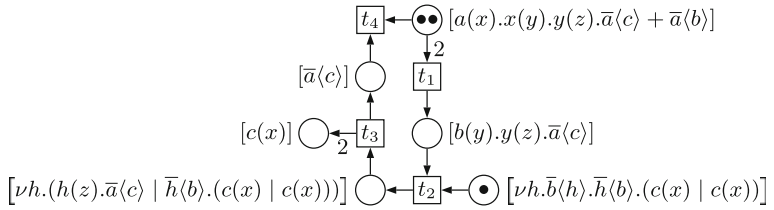

Fig. 4 The structural semantics $\mathcal{N}[\![P]\!]$ of process P in Example 5

The semantics $\mathcal{N}[\![P]\!]$ is depicted in Fig. 4. To begin with, we compute all reachable fragments. They are given by the reaction sequence

$$\begin{aligned}
& a(x).x(y).y(z).\overline{a}\langle c\rangle + \overline{a}\langle b\rangle \mid a(x).x(y).y(z).\overline{a}\langle c\rangle + \overline{a}\langle b\rangle \mid \\
& \nu h.\overline{b}\langle h\rangle.\overline{h}\langle b\rangle.(c(x) \mid c(x)) \\
\to\ & b(y).y(z).\overline{a}\langle c\rangle \mid \nu h.\overline{b}\langle h\rangle.\overline{h}\langle b\rangle.(c(x) \mid c(x)) \\
\to\ & \nu h.(h(z).\overline{a}\langle c\rangle \mid \overline{h}\langle b\rangle.(c(x) \mid c(x))) \\
\to\ & \overline{a}\langle c\rangle \mid c(x) \mid c(x).
\end{aligned}$$

All processes in this reaction sequence are in restricted form. We take the fragments as the set of places:

$$\begin{aligned}
& \{[a(x).x(y).y(z).\overline{a}\langle c\rangle + \overline{a}\langle b\rangle], [b(y).y(z).\overline{a}\langle c\rangle], [\nu h.\overline{b}\langle h\rangle.\overline{h}\langle b\rangle.(c(x) \mid c(x))], \\
& \quad [\nu h.(h(z).\overline{a}\langle c\rangle \mid \overline{h}\langle b\rangle.(c(x) \mid c(x)))], [\overline{a}\langle c\rangle], [c(x)]\}.
\end{aligned}$$

Fragment $F_1 = a(x).x(y).y(z).\overline{a}\langle c\rangle + \overline{a}\langle b\rangle$ reacts with a structurally congruent fragment to $F_2 = b(y).y(z).\overline{a}\langle c\rangle$. This yields $t_1 = ([F_1 \mid F_1], [F_2])$. Both processes, $F_1 \mid F_1$ and F_2, are in restricted form. Thus, the decompositions are defined by $(dec\,(F_1 \mid F_1))\,([F_1]) := 2$, $(dec\,(F_1 \mid F_1))\,([F]) := 0$ otherwise and $(dec\,(F_2))\,([F_2]) := 1$, $(dec\,(F_2))\,([F]) := 0$ otherwise. This explains the arc weights.

Consider $F_3 = \nu h.\overline{b}\langle h\rangle.\overline{h}\langle b\rangle.(c(x) \mid c(x))$. It passes the restricted name h to $F_2 = b(y).y(z).\overline{a}\langle c\rangle$. This results in $\nu h.(h(z).\overline{a}\langle c\rangle \mid \overline{h}\langle b\rangle.(c(x) \mid c(x))) = F_4$. The processes $h(z).\overline{a}\langle c\rangle$ and $\overline{h}\langle b\rangle.(c(x) \mid c(x))$ share the restricted name h, so F_4 is in restricted form. Transition $t_2 = ([F_2 \mid F_3], [F_4])$ models the communication. It demonstrates how the scope of restricted names influences our Petri net semantics. A pair of processes is represented by one token on place $[F_4] = [\nu h.(h(z).\overline{a}\langle c\rangle \mid \overline{h}\langle b\rangle.(c(x) \mid c(x)))]$. All semantics known from the literature (cf. Sect. 7) represent the processes $h(z).\overline{a}\langle c\rangle$ and $\overline{h}\langle b\rangle.(c(x) \mid c(x))$ inside F_4 by separate places, each carrying a token.

Fragment $F_4 = \nu h.(h(z).\overline{a}\langle c\rangle \mid \overline{h}\langle b\rangle.(c(x) \mid c(x)))$ performs an internal reaction. The two processes it consists of communicate on the restricted channel h. The fragment reacts to $Q = \overline{a}\langle c\rangle \mid c(x) \mid c(x)$. By definition, we get the transition $t_3 = ([F_4], [Q])$. There is an arc weighted one from place $[F_4]$ to t_3. The process $Q = \overline{a}\langle c\rangle \mid c(x) \mid c(x)$ is in restricted form. Its decomposition is $dec\,(Q)$ with $(dec\,(Q))\,([\overline{a}\langle c\rangle]) = 1$, $(dec\,(Q))\,([c(x)]) = 2$, and $(dec\,(Q))\,([F]) = 0$ otherwise. The transition shows how fragments consisting of several processes break up if restricted names are forgotten.

The definition of the set of transitions does not take the overall process behaviour into account. The Petri net may contain transitions that are never enabled. Transition t_4 illustrates this fact. The fragments $F_1 = a(x).x(y).y(z).\overline{a}\langle c\rangle + \overline{a}\langle b\rangle$ and $\overline{a}\langle c\rangle$ react to $G = c(y).y(z).\overline{a}\langle c\rangle$. This results in $t_4 = ([F_1 \mid \overline{a}\langle c\rangle], [G])$. The transition is never executed since

the reaction is not possible in the process P. Therefore G is no reachable fragment. In this case, $(dec\,(G))\,([F]) = 0$ for all places $[F]$, so transition t_4 has no places in its postset. Fragment G with $(dec\,(G))\,([G]) = 1$ is not considered.

The process P is in restricted form. The initial marking is given by the decomposition of P, $dec\,(P)$, with $(dec\,(P))\,([a(x).x(y).y(z).\overline{a}\langle c\rangle + \overline{a}\langle b\rangle]) = 2$, $(dec(P))([\nu h.\overline{b}\langle h\rangle.\overline{h}\langle b\rangle.(c(x) \mid c(x))]) = 1$, and $(dec\,(P))\,([F]) = 0$ otherwise. ◇

The computation of transition t_4 in Example 5 indicates that the set of places determines the size of the Petri net.

Lemma 7 *The Petri net $\mathcal{N}[\![P]\!]$ of a process $P \in \mathcal{P}$ is finite if and only if the set of places in $\mathcal{N}[\![P]\!]$ is finite.*

Formally, the lemma follows with the image-finiteness of the reaction relation. Example 5 also reveals that a communication between two fragments requires a public channel. The Petri net of a closed process without public names has transitions of the form $([F], [Q])$, only.

Lemma 8 *If $P \in \mathcal{CP}$ then $\mathcal{N}[\![P]\!]$ is a* communication-free *Petri net, i.e., for all $t \in T : |{}^\bullet t| = 1$ and $\lambda(s,t) \leq 1$ for all $s \in S$.*

To ensure that our semantics is a suitable representation of π-Calculus processes, we show that we can retrieve all information about a process and its reactions from the semantics. To relate a marking in the Petri net $\mathcal{N}[\![P]\!]$ and a process, we define the function *retrieve* : $Reach\,(\mathcal{N}[\![P]\!]) \to \mathcal{P}/_{\equiv}$. It constructs a process from a marking by composing (1) the fragments that are marked in parallel (2) as often as demanded by the marking.

Definition 13 (*retrieve* : $Reach\,(\mathcal{N}[\![P]\!]) \to \mathcal{P}/_{\equiv}$) Given a process $P \in \mathcal{P}$, the function *retrieve* : $Reach\,(\mathcal{N}[\![P]\!]) \to \mathcal{P}/_{\equiv}$ associates with every marking reachable in the structural semantics, $M \in Reach\,(\mathcal{N}[\![P]\!])$, a process class $[Q] \in \mathcal{P}/_{\equiv}$ as follows:

$$retrieve\,(M) := [\Pi_{[F]\in supp(M)} \Pi^{M([F])} F].$$ ◇

The transition systems of a π-Calculus process and its Petri net semantics are isomorphic. Furthermore, the states in both transition systems correspond using the retrieve function.

Theorem 1 (Retrievability) *The transition systems of a process $P \in \mathcal{P}$ and that of its structural semantics $\mathcal{N}[\![P]\!]$ are isomorphic. The isomorphism f : $Reach\,(P)/_{\equiv} \to Reach\,(\mathcal{N}[\![P]\!])$ maps $[Q]$ to $dec\,(\nu(Q))$. The function retrieve is the inverse of f, i.e., a process is reconstructed from a marking by retrieve $(f([Q])) = [Q]$.*

The function f is well-defined due to Proposition 1 and Lemma 5. To prove the theorem we show that *retrieve* is the inverse of f and that f is an isomorphism between the transition systems, i.e., f maps the initial process to the initial marking, f is bijective, and f is a strong graph homomorphism. By definition, a strong graph homomorphism demands that the transition $[P_1] \to [P_2]$ exists in the transition system of the process P if and only if the transition $f([P_1]) \to f([P_2])$ exists in the transition system of the Petri net $\mathcal{N}[\![P]\!]$. Lemma 9 eases the proof of this equivalence as it shows that taking transitions in the Petri net is closely related with performing reactions in the process. The lemma is proven directly via a chain of equivalences. A similar result, stated as Lemma 16 in the appendix, holds for transitions $([F_1 \mid F_2], [Q])$.

Lemma 9 *Consider the function* $f : Reach(P)/_{\equiv} \to Reach(\mathcal{N}[\![P]\!])$ *from Theorem 1. For* $M_1 = f([P_1])$ *and* $M_2 = f([P_2])$ *we have:*

$$\exists([F],[Q]) \in T : M_1(s) \geq \lambda(s,([F],[Q])), \text{ for all } s \in {}^\bullet([F],[Q])$$
$$\text{and } M_2(s) = M_1(s) - \lambda(s,([F],[Q])) + \lambda(([F],[Q]),s), \text{ for all } s \in S$$

if and only if

$$\exists F \in Frag(\nu(Reach(P))), Q, P' \in \mathcal{P} : F \to Q$$
$$\text{and } P_1 \equiv F \mid P' \text{ and } P_2 \equiv Q \mid P'.$$

According to Theorem 1, the semantics preserves up to structural congruence all information about a process. In fact, it describes the process as precise as structural congruence.

Proposition 2 (Full abstraction) *Let* $P, Q \in \mathcal{P}$. *Then* $P \equiv Q$ *if and only if* $\mathcal{N}[\![P]\!] = \mathcal{N}[\![Q]\!]$.

The definition of the structural semantics is declarative as it refers to the set of all reachable fragments and adds transitions where appropriate. In the following section, we comment on the implementation.

3.4 Implementation issues

We implemented the translation in the tool PETRUCHIO [38,39]. Since the set of places in the Petri net $\mathcal{N}[\![P]\!]$ is based on the set of reachable fragments $Frag(\nu(Reach(P)))$ and since the set of reachable processes $Reach(P)$ is defined inductively, our algorithm computes the Petri net $\mathcal{N}[\![P]\!]$ inductively as follows. We determine a sequence of nets $N_0, N_1, N_2, \ldots$ with $N_k = (S_k, T_k, \lambda_k, M_0)$. The initial net is $N_0 = (Frag(\nu(P))/_{\equiv}, \emptyset, \emptyset, dec(\nu(P)))$, i.e., the places are the fragments in the initial process, $S_0 = Frag(\nu(P))/_{\equiv}$, the initial transition set is empty, $T_0 = \emptyset$, and so is the initial weight function, $\lambda_0 = \emptyset$. The initial marking is $M_0 = dec(\nu(P))$.

Assume we computed the net $N_k = (S_k, T_k, \lambda_k, M_0)$. For every place $[F] \in S_k$ that has an internal reaction, i.e., $F \to Q$, we add a transition $t = ([F],[Q])$. Similarly, for two places $[F_1], [F_2] \in S_k$ that are (1) simultaneously markable[1] and (2) able to communicate, i.e., $F_1 \mid F_2 \to Q$ for some Q, we add a transition $t' = ([F_1 \mid F_2],[Q])$. This yields the new transition set T_{k+1}. The postset of a transition t (or t') is the set of fragments in the restricted form of process Q, $Frag(\nu(Q))/_{\equiv}$. The new fragments are added to S_k giving the new set of places S_{k+1}. We get the new weight function λ_{k+1} from λ_k by adding arcs between the new transitions and the (old and new) places according to Definition 12. The initial marking M_0 does not change. The computation stops if there are no more transitions to add, i.e., $N_{k+1} = N_k$. We then have $N_k = \mathcal{N}[\![P]\!]$ as we computed exactly the reachable fragments.[2]

The critical issue in the implementation of the semantics is to determine the places $[F_1], [F_2] \in S_k$, which can be marked simultaneously. We solve the problem by computing the well-known *coverability graph* $Cov(N_k)$ of the nets N_k. The coverability graph is an abstracted state space preserving the information whether a marking is reachable which is greater than or equal to a given marking [37]. To avoid recomputing $Cov(N_k)$ for every net N_k, Tim Strazny showed that the coverability graph $Cov(N_{k+1})$ is an extension of $Cov(N_k)$ [39].

[1] In case $[F_1] = [F_2]$, the place needs to be markable by two tokens.

[2] More precisely, N_k equals $\mathcal{N}[\![P]\!]$ without dead transitions like t_4 in Example 5.

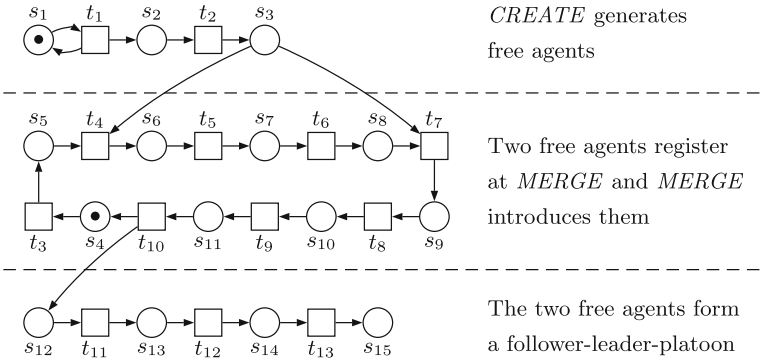

Fig. 5 $\mathcal{N}[\![\mathit{CREATE}\lfloor \mathit{CFA}\rfloor \mid \mathit{MERGE}\lfloor \mathit{CFA}\rfloor]\!]$

To make use of today's multi-core computers, the compiler is implemented in a dual-threaded architecture. The first thread updates the coverability graph and the second thread computes the Petri net as described above.

Note that it is not necessary to compute the coverability graph if P is a closed process, $P \in \mathcal{CP}$. According to Lemma 8, $\mathcal{N}[\![P]\!]$ is a communication-free Petri net, i.e., it only contains transitions $([F], [Q])$, which depend on a single place that is markable.

Due to memory limitations, the coverability graph of a subnet of $\mathcal{N}[\![P]\!]$ may not be computable in practice. In this case, we add the transition $([F_1 \mid F_2], [Q])$ if the places $[F_1]$ and $[F_2]$ can communicate, i.e., $F_1 \mid F_2 \rightarrow Q$, regardless of whether $[F_1]$ and $[F_2]$ can be marked simultaneously. This results in a Petri net $\mathcal{N}_{NoCov}[\![P]\!]$ which subsumes $\mathcal{N}[\![P]\!]$, i.e., $\mathcal{N}[\![P]\!] = (S, T, \lambda, M_0)$ and $\mathcal{N}_{NoCov}[\![P]\!] = (S', T', \lambda', M_0')$ with $S \subseteq S'$, $T \subseteq T'$, $\lambda \subseteq \lambda'$, and $M_0 = M_0'$. The transition systems of $\mathcal{N}[\![P]\!]$ and $\mathcal{N}_{NoCov}[\![P]\!]$ are still isomorphic. The reason is that a transition $([F_1 \mid F_2], [Q])$, which is added although the places $[F_1]$ and $[F_2]$ are not simultaneously markable, is never enabled. A negative effect of this inaccuracy is that $\mathcal{N}_{NoCov}[\![P]\!]$ is often much larger than $\mathcal{N}[\![P]\!]$ and may even become infinite although $\mathcal{N}[\![P]\!]$ is finite.

The following Sect. 4 demonstrates the application of our semantics in the automatic verification of the car platoon case study. We show (1) how to infer interesting properties of different kinds (2) using efficient algorithms.

4 Application of the semantics

First, we investigate the occurrence numbers of processes in the car platoon system. Then we turn to topological properties, asking for connections between processes. By proving a temporal property, we conclude the section. Throughout, we aim at verification techniques that exploit the graph structure of the Petri net instead of state space computations in order to circumvent the state explosion problem.

Example 6 (Structural semantics of the car platoon case study) The Petri net semantics of $\mathit{CREATE}\lfloor \mathit{CFA}\rfloor \mid \mathit{MERGE}\lfloor \mathit{CFA}\rfloor$ is depicted in Fig. 5. We explain the meanings of places and transitions in more detail.

Initially, the processes $\mathit{CREATE}\lfloor \mathit{CFA}\rfloor$ and $\mathit{MERGE}\lfloor \mathit{CFA}\rfloor$ are present. The corresponding places $s_1 = [\mathit{CREATE}\lfloor \mathit{CFA}\rfloor]$ and $s_4 = [\mathit{MERGE}\lfloor \mathit{CFA}\rfloor]$ are marked. With transition t_1,

the process $CREATE\lfloor CFA\rfloor$ generates free agents $s_2 = [FA\lfloor CFA\rfloor]$. Transition t_2 represents a call to the process identifier FA, which yields the place

$$s_3 = [\nu id, ca, rq.\overline{CFA}\langle id\rangle.\overline{id}\langle ca\rangle.\overline{id}\langle rq\rangle.choice].$$

We use *choice* as a shortcut, which is replaced by the following choice composition to obtain the full definition of s_3:

$$ca(rqnl).REQ\lfloor id, rqnl\rfloor + rq(nf).\overline{nf}\langle id\rangle.LD\lfloor id, nf\rfloor.$$

Shortcuts improve the readability of processes, they are not part of the π-Calculus syntax. The call $MERGE\lfloor CFA\rfloor$, represented by transition t_3, gives

$$s_5 = [CFA(id_x).id_x(ca_x).id_x(rq_x).reg_y.\overline{ca_y}\langle rq_x\rangle.MERGE\lfloor CFA\rfloor],$$

where reg_y abbreviates

$$CFA(id_y).id_y(ca_y).id_y(rq_y).$$

With t_4 the first free agent passes its *id* to the *MERGE* process. Therefore, the transition consumes a token for a free agent from s_3 and the token for the *MERGE* process from s_5 and produces a token on s_6, i.e., a fragment that contains a free agent and the *MERGE* process. With t_5 from s_6 to s_7 and t_6 from s_7 to s_8 the free agent continues to pass its *ca* and *rq* channels:

$$\begin{aligned}
s_6 &= \big[\nu id.\,\big(\nu ca, rq.\overline{id}\langle ca\rangle.\overline{id}\langle rq\rangle.choice \\
&\qquad \mid\ id(ca_x).id(rq_x).reg_y.\overline{ca_y}\langle rq_x\rangle.MERGE\lfloor CFA\rfloor\big)\big] \\
s_7 &= [\nu id.\,\big(\nu ca, rq.\overline{id}\langle rq\rangle.choice \mid id(rq_x).reg_y.\overline{ca_y}\langle rq_x\rangle.MERGE\lfloor CFA\rfloor\big)] \\
s_8 &= \big[\nu rq.\,\big(\nu id, ca.choice \mid reg_y.\overline{ca_y}\langle rq\rangle.MERGE\lfloor CFA\rfloor\big)\big].
\end{aligned}$$

The registration of the second free agent, which yields the places s_9, s_{10}, and s_{11} given below, is similar. We use α-conversion to rename the *id*, *ca*, and *rq* channels of the first free agent to id_1, ca_1, and rq_1. The shortcuts $choice_1$ and $choice_2$ correspond to *choice* with those names changed accordingly:

$$\begin{aligned}
s_9 &= \big[\nu id_2.\,\big(\nu rq_1.\,\big(\nu id_1, ca_1.choice_1 \mid id_2(ca_y).id_2(rq_y).\overline{ca_y}\langle rq_1\rangle.MERGE\lfloor CFA\rfloor\big) \\
&\qquad \mid\ \nu ca_2, rq_2.\overline{id_2}\langle ca_2\rangle.\overline{id_2}\langle rq_2\rangle.choice_2\big)\big] \\
s_{10} &= \big[\nu ca_2, id_2.\,\big(\nu rq_1.\,\big(\nu id_1, ca_1.choice_1 \mid id_2(rq_y).\overline{ca_2}\langle rq_1\rangle.MERGE\lfloor CFA\rfloor\big) \\
&\qquad \mid\ \nu rq_2.\overline{id_2}\langle rq_2\rangle.choice_2\big)\big] \\
s_{11} &= \big[\nu ca_2.\big(\nu rq_1.\big(\nu id_1, ca_1.choice_1 \mid \overline{ca_2}\langle rq_1\rangle.MERGE\lfloor CFA\rfloor\big) \\
&\qquad \mid\ \nu id_2, rq_2.(ca_2(rqnl).REQ\lfloor id_2, rqnl\rfloor + \ldots)\big)\big].
\end{aligned}$$

In s_{11}, the *MERGE* process is ready to pass the rq_1 channel of the leading car to the second free agent. Since it uses the ca_2 channel, we say that the *MERGE* process *sends a car ahead message*. Afterwards, the process $MERGE\lfloor CFA\rfloor$ forgets the restricted names of both free agents and the fragment is split up. The free agent that receives the car ahead message becomes a $REQ\lfloor id_2, rq_1\rfloor$ process in s_{12}. With transition t_{11} from s_{12} to s_{13}, *REQ* is replaced by its defining process:

$$\begin{aligned}
s_{12} &= \big[\nu rq_1.\big(\nu id_1, ca_1.choice_1 \mid \nu id_2.REQ\lfloor id_2, rq_1\rfloor\big)\big] \\
s_{13} &= \big[\nu rq_1.\big(\nu id_1, ca_1.(\ldots + rq_1(nf).\overline{nf}\langle id_1\rangle.LD\lfloor id_1, nf\rfloor) \\
&\qquad \mid\ \nu id_2.\overline{rq_1}\langle id_2\rangle.id_2(nl).FL\lfloor id_2, nl\rfloor\big)\big].
\end{aligned}$$

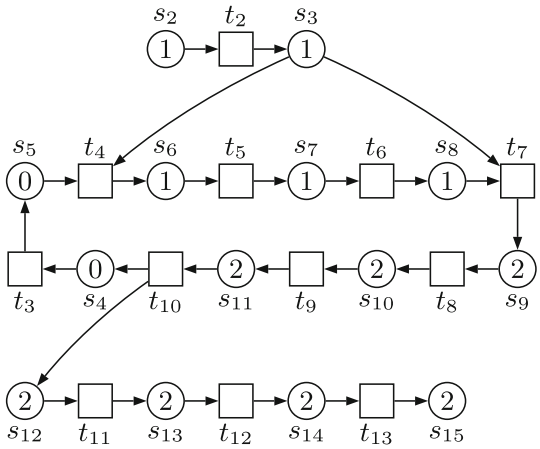

Fig. 6 Subnet of $\mathcal{N}[\![CREATE\lfloor CFA\rfloor \mid MERGE\lfloor CFA\rfloor]\!]$ with a constant number of cars. The place inscriptions form an S-Invariant I

In s_{13}, the second free agent issues a request to merge with the leading car. Transition t_{12} from s_{13} to s_{14} models the acceptance of this request as it reflects the communication of both cars on the rq_1 channel:

$$s_{14} = \left[\nu id_2.\left(\nu id_1.\overline{id_2}\langle id_1\rangle.LD\lfloor id_1, id_2\rfloor \mid id_2(nl).FL\lfloor id_2, nl\rfloor\right)\right]$$
$$s_{15} = \left[\nu id_1, id_2.\left(LD\lfloor id_1, id_2\rfloor \mid FL\lfloor id_2, id_1\rfloor\right)\right].$$

With t_{13} from s_{14} to s_{15}, the now leader passes its id_1 channel to the follower, which yields the car platoon in s_{15}. ◇

4.1 Occurrence number properties

If we consider a reaction sequence where the number of cars stays constant, we expect a linear relationship between the number of free agents and the number of follower-leader-platoons. Each follower-leader-platoon created in the execution should correspond to two free agents beforehand. This relationship exists, we prove it using S-Invariants.

The unmarked subnet of $\mathcal{N}[\![CREATE\lfloor CFA\rfloor \mid MERGE\lfloor CFA\rfloor]\!]$ depicted in Fig. 6 consists of all places except the process $s_1 = [CREATE\lfloor CFA\rfloor]$, which recursively generates free agents. Thus, the number of cars stays constant. To relate the number of free agents and the number of follower-leader-platoons, we construct a relation between the markings of $s_2 = [FA\lfloor CFA\rfloor]$ and $s_{15} = [\nu id_1, id_2.(LD\lfloor id_1, id_2\rfloor \mid FL\lfloor id_2, id_1\rfloor)]$. We use an S-invariant I that covers both places, i.e., $I(s_2) \neq 0 \neq I(s_{15})$. The place inscriptions in Fig. 6 give such an invariant. By the fundamental property of S-Invariants it holds for all markings M and M' of the subnet with $M \rightarrow^* M'$:

$$I^t \cdot M' = I^t \cdot M$$
$$\Leftrightarrow \Sigma_{i=2}^{15} I(s_i)M'(s_i) = \Sigma_{i=2}^{15} I(s_i)M(s_i).$$

If we assume that all merging activities are finished in M and in M', i.e., there are only tokens on the places $s_2 = [FA\lfloor CFA\rfloor]$, $s_4 = [MERGE\lfloor CFA\rfloor]$, and $s_{15} = [\nu id_1, id_2.(LD\lfloor id_1, id_2\rfloor \mid FL\lfloor id_2, id_1\rfloor)]$, then the equation implies:

$$M'(s_2) + 0M'(s_4) + 2M'(s_{15}) = M(s_2) + 0M(s_4) + 2M(s_{15})$$
$$\Leftrightarrow \Delta_{M,M'}(s_{15}) = -1/2\Delta_{M,M'}(s_2),$$

where $\Delta_{M,M'}(s) := M'(s) - M(s)$. This means for every token added on $s_{15} = [vid_1, id_2.(LD\lfloor id_1, id_2\rfloor \mid FL\lfloor id_2, id_1\rfloor)]$ two free agents are removed from $s_2 = [FA\lfloor CFA\rfloor]$. Since $[vid_1, id_2.(LD\lfloor id_1, id_2\rfloor \mid FL\lfloor id_2, id_1\rfloor)]$ represents two processes and no processes are created, we conclude on process level that every free agent removed in a reaction sequence from process P to P' is a follower or a leader in P'.

To sum up, consider $P \in Reach\,(CREATE\lfloor CFA\rfloor \mid MERGE\lfloor CFA\rfloor)$ and $P' \in Reach\,(P)$, where the restricted forms of P and P' consist of fragments $FA\lfloor CFA\rfloor$, $MERGE\lfloor CFA\rfloor$, and $vid_1, id_2.(LD\lfloor id_1, id_2\rfloor \mid FL\lfloor id_2, id_1\rfloor)$ only. Let P' be reachable without $CREATE\lfloor CFA\rfloor \rightarrow FA\lfloor CFA\rfloor \mid CREATE\lfloor CFA\rfloor$. Then the following result holds.

Result 1 *The number of follower-leader-platoons added in P' is half the number of free agents removed in P'. Every free agent removed is a follower or a leader.*

4.2 Topological properties

In Sect. 3.1, we discussed the interpretation of π-Calculus processes as bipartite graphs, where restricted names connect the sequential processes that share them. Inspired by this graph interpretation, we consider *connectedness properties*. We say that two sequential processes $Q, Q' \in \mathcal{S}(P)$ are *directly connected* in the process P if they share a free name, i.e., $fn\,(Q) \cap fn\left(Q'\right) \neq \emptyset$. Assume Q occurs in every reachable process only in fragments where all of its free names are restricted. Then the direct connectedness property can be established for all reachable processes by inspecting all places in the Petri net, i.e., without taking behavioural information into account. Note that the assumption always holds for closed processes $P \in \mathcal{CP}$.

Result 2 *In every reachable process, a follower is directly connected with a leader, i.e., a process that contains $FL\lfloor id_x, id_y\rfloor$ also contains $LD\lfloor id_y, id_z\rfloor$.*

Proof Only fragment $[vid_1, id_2.(LD\lfloor id_1, id_2\rfloor \mid FL\lfloor id_2, id_1\rfloor)]$ contains a follower $FL\lfloor id_2, id_1\rfloor$ and id_1 is the identifier of a leader. □

The proof shows more. There is no situation, in which a follower knows a leader but the leader does not know the follower, i.e., $FL\lfloor id_2, id_1\rfloor$ implies $LD\lfloor id_1, id_z\rfloor$ with $id_z = id_2$. With the argument that a leader is only present in the mentioned fragment we also conclude that a leader is always properly connected, where properly means directly connected with a follower, not with another leader or a free agent.

We say that two sequential processes $Q, Q' \in \mathcal{S}(P)$ are *connected* in the process P, if they are in the transitive closure of the direct connection relation, i.e., there are sequential processes $Q_1, \ldots, Q_n \in \mathcal{S}(P)$ so that $fn\,(Q) \cap fn\,(Q_1) \neq \emptyset$, $fn\,(Q_i) \cap fn\,(Q_{i+1}) \neq \emptyset$, and $fn\,(Q_n) \cap fn\left(Q'\right) \neq \emptyset$. A connection is necessary for an interaction between Q and Q'. The following situation demonstrates that connections are critical. In s_{11}, a free agent waits for the request channel of its new leader, $ca_2(rqnl).REQ\lfloor id_2, rqnl\rfloor + \ldots$ At the same time it is connected (via $\overline{ca_2}\langle rq_1\rangle.MERGE\lfloor CFA\rfloor$) with another free agent $vid, ca, rq.\overline{CFA}\langle id\rangle.\overline{id}\langle ca\rangle.\overline{id}\langle rq\rangle.choice$. A malicious second free agent could send false information to the first.

Result 3 *In $CREATE\lfloor CFA\rfloor \mid MERGE\lfloor CFA\rfloor$ a process is reachable where $ca_2(rqnl).REQ\lfloor id_2, rqnl\rfloor + \ldots$ and $vid, ca, rq.\overline{CFA}\langle id\rangle.\overline{id}\langle ca\rangle.\overline{id}\langle rq\rangle.choice$ are connected.*

Proof The process $ca_2(rqnl).REQ\lfloor id_2, rqnl\rfloor + \ldots$ only shares its ca_2 channel with the process $\overline{ca_2}\langle rq_1\rangle.MERGE\lfloor CFA\rfloor$ in

$$s_{11} = \left[\nu ca_2.\left(\nu rq_1.\left(vid_1, ca_1.choice_1 \mid \overline{ca_2}\langle rq_1\rangle.MERGE\lfloor CFA\rfloor\right)\right.\right.$$
$$\left.\left. \mid\ vid_2, rq_2.(ca_2(rqnl).REQ\lfloor id_2, rqnl\rfloor + \ldots)\right)\right].$$

The process $\overline{ca_2}\langle rq_1\rangle.MERGE\lfloor CFA\rfloor$ additionally shares the channel CFA with $vid, ca, rq.\ \overline{CFA}\langle id\rangle.\overline{id}\langle ca\rangle.\overline{id}\langle rq\rangle.choice$ in s_3. A process is reachable where all three subprocesses occur in parallel composition iff a marking M is reachable with $M(s_3) > 0$ and $M(s_{11}) > 0$. The coverability graph shows that this is possible. □

4.3 Temporal properties

The topological properties verified in the last section are invariants, i.e., they hold in every reachable process. The property considered in this section talks about more elaborate temporal behaviour. The verification is purely graph theoretic, i.e., it solely relies on the net structure.

Result 4 *For every free agent it holds in every reachable process: if the free agent receives a car ahead message, it will never receive a car ahead message again.*

We briefly discuss the formalisation of the property in a temporal logic. The temporal behaviour (a car ahead message is received at most once) is specified relative to a free agent. Formulae in standard temporal logics like *CTL** refer to a finite set of atomic propositions, which is not suitable for reasoning about an unbounded number of free agents. Instead, the property requires a universal quantifier for restricted names, which covers temporal operators. Logics that support name quantification are presented in [10,7,22].

The proof of Result 4 applies the following more general observation. Consider a non-empty (denoted by $+$) sequence

$$F \mid P \to^{+} F' \mid P' \to Q.$$

If the first and last reaction use the restricted name a (i.e., prefix $\overline{a}\langle b\rangle$) and a is not renamed via α-conversion in the meantime, we say there are *two reactions in the sequence using one restricted name*.

Observation 1 *Consider a reaction sequence with two reactions using one restricted name. Let F and F' be the fragments performing the first and the second reaction. Then there is a directed path in the Petri net from place $[F]$ to $[F']$ so that every fragment on the path has at least one restricted name.*

The idea underlying the observation is that a restricted name a, which is used by F and F' with $F \mid P \to P_1 \to \cdots \to P_n \to F' \mid P'$, is remembered in all processes P_i. When the restricted forms are computed, $\nu(P_i) \equiv \Pi_{j\in J_i} G_j$, one fragment G_j contains the restriction νa. These fragments form the path in the Petri net.

Proof of Result 4 To send a car ahead message to the same agent twice, a directed path in the net is needed from the first fragment sending the message to the second. The only fragment sending a car ahead message is

$$\begin{aligned} s_{11} = \big[\nu ca_2.\big(\nu rq_1.\big(vid_1, ca_1.choice_1 \mid \overline{ca_2}\langle rq_1\rangle.MERGE\lfloor CFA\rfloor\big) \\ \mid\ vid_2, rq_2.(ca_2(rqnl).REQ\lfloor id_2, rqnl\rfloor + \cdots)\big)\big]. \end{aligned}$$

The only cycle starting in s_{11} is $s_{11}.s_4.s_5.s_6.s_7.s_8.s_9.s_{10}$ (cf. Fig. 5). Since place $s_4 = [MERGE\lfloor CFA\rfloor]$ on this cycle does not have a restricted name, no car ahead message is sent repeatedly to the same free agent. □

In the following section, we discuss our verification techniques, and report on first experiments to verify the nets obtained from our translation with standard algorithms from Petri net theory.

4.4 Discussion of the verification approach

In Sect. 4.1, we pruned the net $\mathcal{N}[\![\mathit{CREATE}\lfloor \mathit{CFA}\rfloor \mid \mathit{MERGE}\lfloor \mathit{CFA}\rfloor]\!]$ by hand to exclude an unbounded generation of free agents by $\mathit{CREATE}\lfloor \mathit{CFA}\rfloor$. Since our Petri net semantics is non-compositional, excluding the process $\mathit{CREATE}\lfloor \mathit{CFA}\rfloor$ and computing the Petri net $\mathcal{N}[\![\mathit{MERGE}\lfloor \mathit{CFA}\rfloor]\!]$ does not yield the subnet in Fig. 6. Nevertheless, automating our pruning method should be possible for Petri nets $\mathcal{N}[\![P]\!]$, where P uses an environment process like $\mathit{CREATE}\lfloor \mathit{CFA}\rfloor$ to create new processes.

Our verification algorithms exploit the fact that the places in the Petri net $\mathcal{N}[\![P]\!]$ are the reachable fragments of P. Thus, the ease of verification comes at the expense of a complicated computation of the semantics that determines precisely the reachable fragments. More syntactical Petri net translations like [3,11,17] can be computed more efficiently (cf. Sect. 7 for a discussion of these semantics). But they ask for more expensive analyses. A topological verification problem like the correct connection between follower and leader requires to solve a complicated coverability problem in these translations. In general, such a coverability problem needs to be solved in every analysis while our semantics requires a complicated computation once and then eases the verification. Moreover, in Sect. 7 we discuss that our translation still yields finite place/transition Petri nets where related approaches yield either infinite nets or Turing complete models.

In Sect. 3.4 we mentioned that we may not be able to determine the precise set of reachable fragments in the Petri net $\mathcal{N}[\![P]\!]$ due to memory limitations. In this case, we compute an overapproximating Petri net $\mathcal{N}_{\mathit{NoCov}}[\![P]\!]$, which subsumes $\mathcal{N}[\![P]\!]$. Our verification techniques that rely on the knowledge of the reachable fragments are still applicable to $\mathcal{N}_{\mathit{NoCov}}[\![P]\!]$ as follows. If we check the correct connection between a follower and a leader, we inspect all places in $\mathcal{N}_{\mathit{NoCov}}[\![P]\!]$. If the processes are properly connected in $\mathcal{N}_{\mathit{NoCov}}[\![P]\!]$ we can conclude that they are properly connected in $\mathcal{N}[\![P]\!]$. If we find a place $[F]$ in $\mathcal{N}_{\mathit{NoCov}}[\![P]\!]$ where a follower and a leader are not properly connected, we have to check whether $[F]$ is markable in $\mathcal{N}_{\mathit{NoCov}}[\![P]\!]$. It is markable in $\mathcal{N}_{\mathit{NoCov}}[\![P]\!]$ if and only if it is contained in $\mathcal{N}[\![P]\!]$. Thus, if it is markable there is an incorrect connection in $\mathcal{N}[\![P]\!]$ and the property does not hold for the process P. If it is not markable, the place is not contained in $\mathcal{N}[\![P]\!]$. We remove it from $\mathcal{N}_{\mathit{NoCov}}[\![P]\!]$ and repeat the analysis. This approach is a variant of counterexample-guided refinement of an imprecise system representation as proposed in [8].

In [31] we demonstrated that our Petri net semantics works well with efficient standard verification techniques for Petri nets. For the large class of *finite control processes* [10] our translation yields bounded Petri nets with small bounds. These bounded nets can then be analysed with verification tools that implicitly represent a Petri net's state space using so-called unfoldings [18,25,28]. The benchmarks in [31] show that the approach of combining our translation with unfolding-based verification for Petri nets outperforms established π-Calculus verification tools like the MOBILITY WORKBENCH [41] or the HD- AUTOMATA LABORATORY [22], and also related unfolding-based verification techniques for the π-Calculus [26] in terms of memory consumption and runtime.

The performance of Petri net verification tools highly depends on the size of the nets. Our translation yields small nets when the number of processes inside fragments is small or the processes inside fragments tightly interact. For example, the translation of finite state restriction-free processes often yields nets logarithmic in the size of the state space of the process.[3] In the worst case, $\mathcal{N}[\![P]\!]$ represents the state space of a process P. We found out that the

[3] In Sect. 5.2, we show that also infinite state restriction-free processes yield finite Petri nets with our translation.

size of our translation is particularly sensitive to independent reactions inside a fragment. As an example, consider the fragment $\nu a.(\tau.\overline{a}\langle a_1\rangle \mid \ldots \mid \tau.\overline{a}\langle a_n\rangle)$, which yields 2^n places in our translation. Partial-order and Petri net reduction techniques [9,19] are helpful to limit the size of the Petri net, some of which are already implemented in PETRUCHIO.

All automatic verification algorithms we used rely on the finiteness of the Petri net. This motivates our research on conditions for processes to have finite nets under the structural semantics in the following Sect. 5.

5 Structural stationarity

Since the π-Calculus is Turing complete but finite place/transition Petri nets are not, our semantics yields infinite nets for some processes. In this section, we investigate *structural stationarity*. This property characterises the set of processes that are translated into finite nets, i.e., the structural semantics of a process is finite if and only if the process is structurally stationary.

The definition of structural stationarity below refers to the form of all reachable fragments. Therefore, it is hard to prove that a given process has the property. We present two complete characterisations of structural stationarity, which refer to the parallel composition and to the restriction operator, respectively. The first characterisation is helpful to prove that a process is structurally stationary and thus our semantics is suited to finitely represent it. The latter characterisation is based on the novel functions *depth* and *breadth*. It is applied in contraposition (1) to show our semantics does not finitely represent the process but in particular (2) to assess the verification problem. We return to this discussion at the end of Sect. 5.2.

If there are finitely many fragments so that the restricted form of all reachable processes is a parallel composition of those fragments, we call the process structurally stationary. Intuitively, there is a finite number of types of fragments in the system.

Definition 14 (Structural stationarity) A process $P \in \mathcal{P}$ is *structurally stationary* if there is a finite set of fragments such that the fragments of all reachable processes are up to structural congruence included in that set:

$$\exists\{F_1, \ldots, F_n\} : \forall Q \in Reach(P) : \forall F \in Frag(\nu(Q)) : \exists i : F \equiv F_i. \qquad \diamond$$

Lemma 10 states the mentioned equivalence between the finiteness of the semantics and structural stationarity.

Lemma 10 (Finiteness) *The structural semantics $\mathcal{N}[\![P]\!]$ is finite if and only if the process P is structurally stationary.*

In the following Sect. 5.1, we investigate the notion of *derivatives*, a finite set of processes assigned to a process P. All processes reachable from P are created from elements in *derivatives*(P) via parallel composition, restriction, and substitution. The corresponding Proposition 3 is crucial in the proof of the first characterisation of structural stationarity, Theorem 2.

5.1 Derivatives

The idea to construct the set of *derivatives* of a given process P is to recursively remove all prefixes as if they were consumed in communications. If a process identifier K is called, directly in P or indirectly in one of its defining equations, we also add the derivatives of

the process defining K. So, for the process $b(y).\overline{y}\langle b\rangle.K\lfloor a\rfloor$ with $K(x) := \overline{x}\langle x\rangle$ we get the derivatives $b(y).\overline{y}\langle b\rangle.K\lfloor a\rfloor$, $\overline{y}\langle b\rangle.K\lfloor a\rfloor$, $K\lfloor a\rfloor$, and $\overline{x}\langle x\rangle$.

Definition 15 (*derivatives* : $\mathcal{P} \to \mathbb{P}(\mathcal{P})$) To define the derivatives of a process we need the function $der : \mathcal{P} \to \mathbb{P}(\mathcal{P})$, which is given by:

$$\begin{aligned} der(\mathbf{0}) &:= \emptyset & der(P_1 \mid P_2) &:= der(P_1) \cup der(P_2) \\ der(K\lfloor\tilde{a}\rfloor) &:= \{K\lfloor\tilde{a}\rfloor\} & der(\nu a.P) &:= der(P). \\ der(\Sigma_{i\in I\neq\emptyset}\pi_i.P_i) &:= \{\Sigma_{i\in I\neq\emptyset}\pi_i.P_i\} \cup \textstyle\bigcup_{i\in I} der(P_i) \end{aligned}$$

Consider $P \in \mathcal{P}$. The set of *derivatives* of P, *derivatives*(P), is the smallest set so that (1) $der(P) \subseteq$ *derivatives*(P) and (2) if $K\lfloor\tilde{a}\rfloor \in$ *derivatives*(P) then $der(Q) \subseteq$ *derivatives*(P), where $K(\tilde{x}) := Q$. ◇

There are two differences between the derivatives and the processes obtained with the reaction relation. (1) Names y that are replaced by received names when an action $b(y)$ is consumed remain unchanged in the derivatives. (2) Parameters $\tilde{x}$ that are replaced by $\tilde{a}$ when an identifier $K\lfloor\tilde{a}\rfloor$ is called are not replaced in the derivatives. Both shortcomings are corrected by substitutions applied to the free names in the derivatives. Proposition 3 shows that this yields all processes reachable in the system.

Proposition 3 *Let $P \in \mathcal{P}$. Every $Q \in$ Reach(P) and every $F \in$ Frag$(\nu(Q))$ is structurally congruent with $\nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i)$, where $Q_i \in$ derivatives(P) and $\sigma_i : fn(Q_i) \to fn(P) \cup \tilde{a}$.*

The following example gives an intuitive understanding to this technical statement.

Example 7 (Derivatives) Consider $P = \nu b.\overline{a}\langle b\rangle.b(x) \mid a(y).K\lfloor a, y\rfloor$, where $K(a, y) := \overline{y}\langle a\rangle$. The only reaction sequence is

$$\nu b.\overline{a}\langle b\rangle.b(x) \mid a(y).K\lfloor a, y\rfloor \to \nu b.(b(x) \mid K\lfloor a, b\rfloor) \to \nu b.(b(x) \mid \overline{b}\langle a\rangle) \to \mathbf{0}.$$

We compute the set of derivatives:

$$derivatives(P) = \{\overline{a}\langle b\rangle.b(x), b(x), a(y).K\lfloor a, y\rfloor, K\lfloor a, y\rfloor, \overline{y}\langle a\rangle\}.$$

The following congruences show that every reachable fragment can be constructed from the derivatives as stated in Proposition 3. The reachable fragments are depicted to the left, the constructed processes to the right:

$$\begin{aligned} \nu b.\overline{a}\langle b\rangle.b(x) &\equiv \nu b.((\overline{a}\langle b\rangle.b(x))\{a, b/a, b\}) \\ a(y).K\lfloor a, y\rfloor &\equiv (a(y).K\lfloor a, y\rfloor)\{a/a\} \\ \nu b.(b(x) \mid K\lfloor a, b\rfloor) &\equiv \nu b.(b(x)\{b/b\} \mid K\lfloor a, y\rfloor\{a, b/a, y\}) \\ \nu b.(b(x) \mid \overline{b}\langle a\rangle) &\equiv \nu b.(b(x)\{b/b\} \mid \overline{y}\langle a\rangle\{b, a/y, a\}). \end{aligned}$$

The process $\mathbf{0}$ is represented by a parallel composition with an empty index set, i.e., $\Pi_{i\in\emptyset} P_i$. ◇

In the proof of Theorem 2, the finiteness of the set of derivatives is important. There we construct for a given process P a finite set of fragments FG using *derivatives*(P). An application of Proposition 3 then shows that every fragment reachable from P is structurally congruent with a fragment in FG.

Lemma 11 *The set derivatives(P) is finite for all $P \in \mathcal{P}$.*

5.2 Characterisations of structural stationarity

In this section, we provide the two characterisations of structural stationarity mentioned above. To begin with, we prove the characterisation of structural stationarity that refers to the parallel composition operator, i.e., structural stationarity is equivalent to boundedness of all reachable fragments in the number of sequential processes. Several well-known subclasses of π-Calculus are immediately shown to be structurally stationary with an application of this characterisation. Furthermore, the proofs of Theorems 3 and 4 in this paper underline its importance. Formally, the number of sequential processes is defined by counting the non-empty sums and process identifiers, e.g., $number_{|}(\nu b.(b(x) \mid K\lfloor a, b\rfloor)) = 2$. It is straightforward to show that the function is invariant under structural congruence.

Definition 16 ($number_{|} : \mathcal{P} \to \mathbb{N}$) The *number of sequential processes* inside $P \in \mathcal{P}$ is given by $number_{|}(P)$ defined inductively:

$$\begin{aligned} number_{|}(\mathbf{0}) &:= 0 \\ number_{|}(K\lfloor\tilde{a}\rfloor) &:= number_{|}\left(\Sigma_{i\in I\neq\emptyset}\pi_i.P_i\right) := 1 \\ number_{|}(P_1 \mid P_2) &:= number_{|}(P_1) + number_{|}(P_2) \\ number_{|}(\nu a.P) &:= number_{|}(P)\,. \end{aligned}$$ ◇

We state the characterisation via boundedness of $number_{|}$ in Theorem 2. In the proof, boundedness follows immediately from structural stationarity.

Theorem 2 (Characterisation of structural stationarity via |) *A process $P \in \mathcal{P}$ is structurally stationary if and only if there is a bound on the number of sequential processes in all reachable fragments, i.e.,*

$$\exists k_s \in \mathbb{N} : \forall Q \in Reach(P) : \forall F \in Frag(\nu(Q)) : number_{|}(F) \leq k_s.$$

To establish structural stationarity from boundedness in $number_{|}(F)$, we construct a finite set of fragments FG, which includes up to structural congruence every reachable fragment. More precisely, we show that for every process $Q \in Reach(P)$ and every fragment $F \in Frag(\nu(Q))$ there is a fragment $F' \in FG$ with $F' \equiv F$.

We first explain the idea underlying the construction of the fragments in FG and then turn to the technicalities. The set FG is the union of sets FG_i that contain fragments with i sequential processes. To build the fragments in FG_i, we consider processes $\nu\tilde{a}.(\Pi^i_{j=1} Q_j\sigma'_j)$ of the form in Proposition 3, i.e., the Q_j are derivatives of P and the σ'_j are substitutions mapping $fn\left(Q_j\right)$ into $fn(P) \cup \tilde{a}$. We rename the names $\tilde{a}$ to a bounded set of unique names $\tilde{u}_i$, parameterised by the number of processes i. This ensures we only need to consider finitely many substitutions σ_j for every derivative Q_j. We add the restricted form $\nu\left(\nu\tilde{u}_i.(\Pi^i_{j=1} Q_j\sigma_j)\right)$ to FG_i, if it is a fragment.

Let k_s be a bound on the number of sequential processes in all reachable fragments. Lemma 11 gives the finiteness of $derivatives(P)$. Thus the maximum $\max FN := \max\{|fn(Q)| \mid Q \in derivatives(P)\}$ on the number of free names in derivatives exists. Let $\tilde{u}_i := u_1, \ldots, u_{i\cdot\max FN}$ be unique names distinct from the free names in P. For every $i \in \mathbb{N}$ the set FG_i is defined by:

$$\left\{\nu\left(\nu\tilde{u}_i.(\Pi^i_{j=1} Q_j\sigma_j)\right) \mid Q_j \in derivatives(P),\ \sigma_j : fn\left(Q_j\right) \to fn(P) \cup \tilde{u}_i,\right.$$
$$\left.\text{and } \nu\left(\nu\tilde{u}_i.(\Pi^i_{j=1} Q_j\sigma_j)\right) \text{ is a fragment}\right\}.$$

The set FG is the union of all sets FG_i with $i \leq k_s$. This means, the fragments in FG have at most k_s sequential processes:

$$FG := FG_1 \cup \cdots \cup FG_{k_s}.$$

With Proposition 3, it is easy to show that every reachable fragment F is structurally congruent with a fragment $\nu\big(\nu\tilde{u}_{|I|}.(\Pi_{i\in I}\, Q_i\sigma_i)\big)$ in $FG_{|I|}$. To prove the theorem, the inclusion $FG_{|I|} \subseteq FG$ remains to be shown. This follows with $|I| = number_|(F) \leq k_s$, which holds with the invariance of $number_|(-)$ under structural congruence and the assumption. We explain the construction of FG on an example.

Example 8 (FG) Consider $P = \nu b.\overline{a}\langle b\rangle.b(x) \mid a(y).K\lfloor a, y\rfloor$, where $K(a, y) := \overline{y}\langle a\rangle$. In Example 7, we computed the reachable fragments:

$$\nu b.\overline{a}\langle b\rangle.b(x), \quad a(y).K\lfloor a, y\rfloor, \quad \nu b.(b(x) \mid K\lfloor a, b\rfloor), \quad \nu b.(b(x) \mid \overline{b}\langle a\rangle).$$

The number of sequential processes in all reachable fragments is bounded by $k_s = 2$ (which equals, e.g., $number_|(\nu b.(b(x) \mid K\lfloor a, b\rfloor))$. The set FG is therefore defined by $FG = FG_1 \cup FG_2$. The set of derivatives is

$$derivatives(P) = \{\overline{a}\langle b\rangle.b(x), b(x), a(y).K\lfloor a, y\rfloor, K\lfloor a, y\rfloor, \overline{y}\langle a\rangle\}.$$

The maximal number of free names in derivatives is $\max FN = 2$, e.g., given by $|fn(\overline{a}\langle b\rangle.b(x))|$. Thus, FG_1 and FG_2 contain fragments

$$\nu(\nu u_1, u_2.(Q\sigma)) \text{ and } \nu(\nu u_1, \ldots, u_4.(Q_1\sigma_1 \mid Q_2\sigma_2)),$$

respectively, where $Q \in derivatives(P)$ with $\sigma : fn(Q) \to \{u_1, u_2, a\}$ and $Q_j \in derivatives(P)$ with $\sigma_j : fn\big(Q_j\big) \to \{u_1, \ldots, u_4, a\}$, for $j = 1, 2$. As an example, consider $Q = \overline{a}\langle b\rangle.b(x) \in derivatives(P)$. Applying all substitutions $\sigma : \{a, b\} \to \{u_1, u_2, a\}$ yields the following fragments in FG_1, where structurally congruent ones are omitted:

$$\begin{aligned}
\overline{a}\langle b\rangle.b(x)\{a, a/a, b\} &= \overline{a}\langle a\rangle.a(x)\\
\nu u_1.((\overline{a}\langle b\rangle.b(x))\{u_1, u_1/a, b\}) &= \nu u_1.\overline{u_1}\langle u_1\rangle.u_1(x)\\
\nu u_1.((\overline{a}\langle b\rangle.b(x))\{a, u_1/a, b\}) &= \nu u_1.\overline{a}\langle u_1\rangle.u_1(x)\\
\nu u_1.((\overline{a}\langle b\rangle.b(x))\{u_1, a/a, b\}) &= \nu u_1.\overline{u_1}\langle a\rangle.a(x)\\
\nu u_1, u_2.((\overline{a}\langle b\rangle.b(x))\{u_1, u_2/a, b\}) &= \nu u_1, u_2.\overline{u_1}\langle u_2\rangle.u_2(x).
\end{aligned}$$

The reachable fragment $\nu b.\overline{a}\langle b\rangle.b(x)$ is structurally congruent with the element $\nu u_1.\overline{a}\langle u_1\rangle.u_1(x) \in FG_1 \subseteq FG$. ◇

Although the characterisation of structural stationarity in Theorem 2 is semantical in the sense that it refers to all reachable fragments, it has important implications. First, it yields structural stationarity of the syntactic classes of *finite control processes* [10] and *restriction-free processes* [1]. Finite control processes are of the form $\nu\tilde{a}.(P_1 \mid \ldots \mid P_n)$, where the processes P_i do not use the parallel composition operator. Thus, the number of sequential processes in all reachable fragments is bounded by n. In restriction-free processes, defined without the restriction operator, fragments are sequential processes. Thus $number_|(F)$ is bounded by 1. In this paper, we apply the theorem to establish structural stationarity of the syntactic class of *finite handler processes* (cf. Theorem 4). Also the semantic class of *finitary processes* [32,34,36] is shown to be structurally stationary with an application of Theorem 2. Finitary processes require a bound $k_s \in \mathbb{N}$ on the number of sequential processes in every reachable process Q, i.e., $number_|(Q) \leq k_s$. With $number_|(F) \leq$

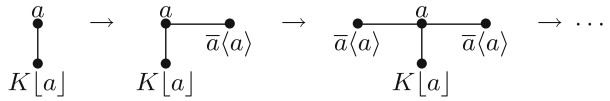

Fig. 7 Reaction sequence illustrating unbounded breadth

$number_{|}(Q)$ for $F \in Frag(\nu(Q))$, k_s binds the number of sequential processes in fragments as well.

As second application, Theorem 2 gives an algorithm to compute the structural semantics without using the coverability graph. The idea is to compute a Petri net $\mathcal{N}_{FG}[\![P]\!]$, which has the set FG in the proof of Theorem 2 as places. Transitions, arcs, and the initial marking are added according to Definition 12. Since FG contains all reachable fragments, the Petri net $\mathcal{N}_{FG}[\![P]\!]$ subsumes $\mathcal{N}[\![P]\!]$. Like for $\mathcal{N}_{NoCov}[\![P]\!]$ in Sect. 3.4, the transition systems of $\mathcal{N}_{FG}[\![P]\!]$ and $\mathcal{N}[\![P]\!]$ are isomorphic but $\mathcal{N}_{FG}[\![P]\!]$ may contain places which are not markable, i.e., which are not in $\mathcal{N}[\![P]\!]$. We plan a prototypical implementation of $\mathcal{N}_{FG}$ in our tool PETRUCHIO. The crucial issue in the implementation is to limit the size of the set FG. Static analysis techniques like [2] may be helpful to solve this problem. We defer the discussion of the work of Bodei et. al. until Sect. 8.

We now turn to the characterisation of structural stationarity that refers to the restriction operator. Its proof is the third application of Theorem 2. We observe that a bounded number of restricted names does not imply structural stationarity. In fact, a process with only one restricted name may not be structurally stationary. Consider $\nu a.K\lfloor a\rfloor$ with $K(x) := \overline{x}\langle x\rangle \mid K\lfloor x\rfloor$. It generates processes sending on the restricted channel a. The reaction sequence

$$\nu a.K\lfloor a\rfloor \to \nu a.(\overline{a}\langle a\rangle \mid K\lfloor a\rfloor) \to \nu a.(\overline{a}\langle a\rangle \mid \overline{a}\langle a\rangle \mid K\lfloor a\rfloor) \to \ldots$$

forms infinitely many fragments that are pairwise not structurally congruent.

In the graph interpretation, depicted in Fig. 7, there is no bound on the number of nodes connected with the node of the restricted name a, i.e., the degree of this node is not bounded. Thus, the degree of the graphs, which is the maximum of the node degrees, is not bounded.

On process level, the degree of a node labelled by a is the number of sequential processes that share the restricted name. In the restricted form, the node degree is reflected by the maximal number of fragments under a restriction $nest_{|}(F)$. For example, the execution above yields $nest_{|}(\nu a.K\lfloor a\rfloor) = 1$ and $nest_{|}(\nu a.(\overline{a}\langle a\rangle \mid K\lfloor a\rfloor)) = 2$. To represent the degree of the graphs, i.e., the maximum of the node degrees, as a function on fragments we search for the widest representation F_B of a fragment F. Widest means that the number of fragments under a restriction in F_B is maximal in the congruence class. The *breadth* of F is then defined by the number of fragments under a restriction in F_B. This definition of breadth fits to the graph interpretation as one can show that the breadth of F equals the degree of $\mathcal{G}[\![F]\!]$.

Definition 17 ($breadth : \mathcal{P}_{\mathcal{F}} \to \mathbb{N}$) The function $nest_{|} : \mathcal{P}_{\mathcal{F}} \to \mathbb{N}$ gives the *maximal number of fragments under a restriction*:

$$nest_{|}\left(\Sigma_{i\in I\neq\emptyset}\pi_i.P_i\right) := nest_{|}(K\lfloor\tilde{a}\rfloor) := 1$$
$$nest_{|}(\nu a.(F_1 \mid \ldots \mid F_n)) := \max\{n, nest_{|}(F_1), \ldots, nest_{|}(F_n)\}.$$

The *breadth* of fragment F is the maximal number of fragments under restrictions in all fragments structurally congruent with F, i.e.,

$$breadth(F) := \max\{nest_{|}(F') \mid F' \equiv F\}. \qquad \diamond$$

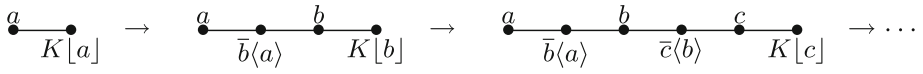

Fig. 8 Reaction sequence illustrating unbounded depth

As it refers to all fragments in a congruence class, the definition of breadth is hard to grasp. We provide a small example that illustrates the definition.

Example 9 (Breadth) Consider $\nu a.L\lfloor a\rfloor$ with $L(x) := \nu b.(\overline{x}\langle b\rangle \mid \overline{x}\langle b\rangle \mid L\lfloor x\rfloor)$. The only reaction sequence is given by

$$\begin{aligned}\nu a.L\lfloor a\rfloor &\to \nu a.(\nu a_1.(\overline{a}\langle a_1\rangle \mid \overline{a}\langle a_1\rangle) \mid L\lfloor a\rfloor)\\ &\to \nu a.(\nu a_1.(\overline{a}\langle a_1\rangle \mid \overline{a}\langle a_1\rangle) \mid \nu a_2.(\overline{a}\langle a_2\rangle \mid \overline{a}\langle a_2\rangle) \mid L\lfloor a\rfloor) \to \dots\end{aligned}$$

After n reactions we have the following fragment $F_D \equiv F_B$:

$$\begin{aligned}F_D &= \nu a.(\Pi_{i=1}^{n}\nu a_i.(\overline{a}\langle a_i\rangle \mid \overline{a}\langle a_i\rangle) \mid L\lfloor a\rfloor)\\ F_B &= \nu a_1.(\dots(\nu a_n.(\nu a.(\Pi_{i=1}^{n}(\overline{a}\langle a_i\rangle \mid \overline{a}\langle a_i\rangle) \mid L\lfloor a\rfloor)))\dots).\end{aligned}$$

We compute $nest_{|}(F_D) = n+1$ and $nest_{|}(F_B) = 2n+1$. In F_B the number of fragments under a restriction is maximal in the congruence class of $F_D \equiv F_B$. So after n reactions we have $breadth(F_D) = breadth(F_B) = nest_{|}(F_B) = 2n+1$. There is no bound on the breadth of the reachable fragments. ◇

Intuitively, the fragment F_B that maximises $nest_{|}$ minimises the scope of the restricted name, which is shared by most sequential processes. We briefly explain the construction of F_B from a given fragment. First, we apply Lemma 2 to compute the process $\nu\tilde{a}.(P_1 \mid \dots \mid P_n)$. One of the names in $\tilde{a}$, say a, is free in most processes P_i. We rearrange the names so that the scope of νa is minimised first, when the restricted form is computed: $\nu\tilde{a}\backslash\{a\}.\nu a.(P_1 \mid \dots \mid P_n)$. The restricted form of this process is the fragment F_B.

Imposing a bound on the breadth of all reachable fragments does not suffice to show structural stationarity. Consider the process $\nu a.K\lfloor a\rfloor$ with $K(x) := \nu a.(\overline{a}\langle x\rangle \mid K\lfloor a\rfloor)$. It generates infinitely many fragments that are pairwise not structurally congruent but have a breadth of two:

$$\nu a.K\lfloor a\rfloor \to \nu a.(\nu b.(\overline{b}\langle a\rangle \mid K\lfloor b\rfloor)) \to \nu a.(\nu b.(\overline{b}\langle a\rangle \mid \nu c.(\overline{c}\langle b\rangle \mid K\lfloor c\rfloor))) \to \dots$$

In the graph interpretation in Fig. 8, the length of the paths that do not repeat name nodes is not bounded. We call those paths *simple*.

On process level, the length of the simple paths is mimicked by the nesting of restrictions $nest_{\nu}(F)$. In the example above, we have $nest_{\nu}(\nu a.K\lfloor a\rfloor) = 1$ and $nest_{\nu}\big(\nu a.(\nu b.(\overline{b}\langle a\rangle \mid K\lfloor b\rfloor))\big) = 2$. To ensure the restrictions contribute to the length of a simple path, we take a representation F_D of the given fragment F where this nesting is minimal. Intuitively, F_D is the flattest representation of F. The *depth* of fragment F is then defined by the nesting of restrictions in this flattest representation F_D. This definition of depth corresponds to the intuitive understanding: the depth of all reachable fragments is bounded if and only if the length of all simple paths in the graph interpretation is bounded [29].

Definition 18 ($depth : \mathcal{P}_{\mathcal{F}} \to \mathbb{N}$) The *nesting of restrictions* in a fragment is given by the function $nest_{\nu} : \mathcal{P}_{\mathcal{F}} \to \mathbb{N}$ defined inductively as follows:

$$\begin{aligned}nest_{\nu}\left(\Sigma_{i\in I\neq\emptyset}\pi_i.P_i\right) &:= nest_{\nu}(K\lfloor\tilde{a}\rfloor) := 0\\ nest_{\nu}(\nu a.(F_1 \mid \dots \mid F_n)) &:= 1 + \max\{nest_{\nu}(F_1), \dots, nest_{\nu}(F_n)\}.\end{aligned}$$

For a fragment F, we define the *depth* to be the minimal nesting of restrictions in all fragments in the congruence class:

$$depth(F) := \min\{nest_{\nu}(F') \mid F' \equiv F\}. \qquad \diamond$$

We continue the investigation of process $\nu a.L\lfloor a \rfloor$ defined in Example 9.

Example 10 (Depth) We observed that all processes reachable via n reactions are structurally congruent with $F_D \equiv F_B$:

$$F_D = \nu a.(\Pi_{i=1}^{n} \nu a_i.(\overline{a}\langle a_i\rangle \mid \overline{a}\langle a_i\rangle) \mid L\lfloor a \rfloor)$$
$$F_B = \nu a_1.(\ldots(\nu a_n.(\nu a.(\Pi_{i=1}^{n}(\overline{a}\langle a_i\rangle \mid \overline{a}\langle a_i\rangle) \mid L\lfloor a \rfloor)))\ldots).$$

The nesting function yields $nest_{\nu}(F_D) = 2$ and $nest_{\nu}(F_B) = n + 1$. Since the nesting of restrictions in F_D is minimal in the congruence class, we have $depth(F_B) = depth(F_D) = nest_{\nu}(F_D) = 2$. So the depth all fragments reachable from $\nu a.L\lfloor a \rfloor$ is bounded by 2. $\diamond$

In a fragment F, there are at most $nest_{|}(F)$ fragments under a restriction. The nesting of restrictions is bounded by $nest_{\nu}(F)$. Thus, F contains at most $nest_{|}(F)^{nest_{\nu}(F)}$ sequential processes. We state this observation in Lemma 12.

Lemma 12 *For all* $F \in \mathcal{P}_{\mathcal{F}} : number_{|}(F) \leq nest_{|}(F)^{nest_{\nu}(F)}$.

The characterisation of structural stationarity in Theorem 3 refers to the restriction operator: a process is structurally stationary if and only if all reachable fragments are bounded in breadth and bounded in depth. While the proof of structural stationarity in Theorem 2 is direct and cumbersome, the application of the theorem yields an elegant proof of Theorem 3. We briefly sketch the proof.

If the process under consideration is bounded in depth by k_d, the nesting of restrictions in the flattest representation F_D of a given fragment F is bounded by k_d since $nest_{\nu}(F_D) = depth(F) \leq k_d$. The number of fragments under a restriction in F_D is bounded by the breadth, which is assumed to be bounded by k_b: $nest_{|}(F_D) \leq breadth(F) \leq k_b$. With Lemma 12 we have a bounded number of sequential processes in F_D and so in F. With Theorem 2 we conclude structural stationarity.

Theorem 3 (Characterisation of structural stationarity via ν) *A process* $P \in \mathcal{P}$ *is structurally stationary if and only if it is bounded in breadth and bounded in depth, i.e.,*

$$\exists k_b \in \mathbb{N} : \forall Q \in Reach(P) : \forall F \in Frag(\nu(Q)) : breadth(F) \leq k_b \text{ and}$$
$$\exists k_d \in \mathbb{N} : \forall Q \in Reach(P) : \forall F \in Frag(\nu(Q)) : depth(F) \leq k_d.$$

The reformulation of the theorem is useful in disproving structural stationarity. A process is not structurally stationary if and only if it is not bounded in breadth or not bounded in depth. Thus, there are two sources of infinity for the structural semantics. We discuss that they are of different quality.

Consider $\nu a.L\lfloor a \rfloor$ with $L(x) := \nu b.(\overline{x}\langle b\rangle \mid \overline{x}\langle b\rangle \mid L\lfloor x \rfloor)$ in Examples 9 and 10. Removing the restriction νa gives the process $L\lfloor a \rfloor$. The semantics $\mathcal{N}[\![L\lfloor a \rfloor]\!]$ is depicted in Fig. 9. In an execution, the number of tokens on $[\nu b.(\overline{a}\langle b\rangle \mid \overline{a}\langle b\rangle)]$ is not bounded. We conclude that $\nu a.L\lfloor a \rfloor$ is not bounded in breadth. With Theorem 3 the process is not structurally stationary.

The example suggests that processes, which are bounded in depth but not bounded in breadth, are not Turing complete. In fact, we have recently shown that termination is decidable for processes of bounded depth in the presence of unbounded breadth [29]. Vice versa,

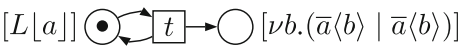

Fig. 9 Identifying and handling unboundedness in breadth

in [1] an encoding of counter machines into the π-Calculus is presented. In our terminology, the construction relies on unboundedness in depth. The breadth is bounded.

The present section identified the properties that lead to finiteness and infinity of the structural semantics. In the following Sect. 6, we apply these insights to design a syntactic class of processes. The goal is to model client-server architectures like the car platoon system. The main theorem states that all processes in the new class are structurally stationary, i.e., finitely represented under the structural semantics. Theorem 2 is a helpful tool in the proof.

6 Finite handler processes

We start with a sketch of the idea underlying our syntactic class of *finite handler processes*. Afterwards, we turn to the definition. A *handler* process listens on a set of channels represented by *distinguished public names*. The analogy is a server located at an IP address listening on a set of ports. To register at a handler, *participant* processes send a restricted name over one of the distinguished channels. This mimics a client that passes its own IP address when contacting a server. The handler introduces the participants to each other and by this creates fragments. No protocol is specified for finite handler processes. The crucial restriction in the communication is that handler processes receive finitely many messages from participants per session. In particular, only finitely many participants can register at a handler. At some point, the handler loses connection to the fragment and restarts listening on the distinguished channels. This ensures boundedness of the fragments in the number of processes.

The set of *distinguished public names* $\mathcal{N}_\mathcal{P}$ is a subset of the set of names, $\mathcal{N}_\mathcal{P} \subseteq \mathcal{N}$. We use the letter p to refer to names in $\mathcal{N}_\mathcal{P}$, whereas a, b, x, y refer to names in $\mathcal{N} \backslash \mathcal{N}_\mathcal{P}$. To syntactically detect the use of distinguished public names we define *receiving prefixes* π^R. They differ from prefixes π in the ability to receive on distinguished names, $\pi^R = p(x)$. An arbitrary prefix π^{RS} also allows for *sending* on distinguished names, $\pi^{RS} = \overline{p}\langle x \rangle$:

$$\pi ::= \overline{x}\langle y \rangle \mid x(y) \mid \tau \qquad \pi^R ::= \pi \mid p(x) \qquad \pi^{RS} := \pi^R \mid \overline{p}\langle x \rangle.$$

The formalisation of *participants* requires some explanation. A participant is a process of the form

$$\nu \tilde{a}.\left(\Sigma_{i \in I}\, \overline{p_i}\langle a_i \rangle.SQ_i\right).$$

To register at a handler process, it sends a name $a_i \in \mathcal{N} \backslash \mathcal{N}_\mathcal{P}$ over a distinguished channel $p_i \in \mathcal{N}_\mathcal{P}$ using the choice composition $\Sigma_{i \in I}\, \overline{p_i}\langle a_i \rangle$. The side condition that all free names in a participant are in $\mathcal{N}_\mathcal{P}$ ensures a_i is restricted, i.e., $a_i \in \tilde{a}$. After the registration, a participant is a process SQ, which is sequential in the sense that it does not contain the parallel composition operator. Hence, communications with participants do not increase the number of sequential processes within fragments. Furthermore, SQ processes do not use distinguished channels. So a participant registers at precisely one handler process.

Definition 19 (Participants) The set of *participants* $\mathcal{P}_{\mathcal{PT}}$ with typical elements $PT \in \mathcal{P}_{\mathcal{PT}}$ is defined inductively by

$$SQ ::= \Sigma_{i\in I}\pi_i.SQ_i \mid \nu a.SQ \mid K_S\lfloor\tilde{a}\rfloor$$
$$PT ::= \nu\tilde{a}.\left(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i\right) \mid K_P\lfloor\tilde{p}\rfloor \mid PT_1 \mid PT_2,$$

where $fn\left(\nu\tilde{a}.\left(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i\right)\right) \subseteq \mathcal{N}_{\mathcal{P}}$. An identifier K_S is defined by an SQ process and an identifier K_P is defined by a process $PT \in \mathcal{P}_{\mathcal{PT}}$. ◇

The car platoon case study turns out to be a finite handler process. Free agents are participants.

Example 11 (Participant) If we choose $\mathcal{N}_{\mathcal{P}} = \{CFA\}$ as the set of distinguished names, the process $FA\lfloor CFA\rfloor$ is a participant. It corresponds to $K_P\lfloor\tilde{p}\rfloor$ in Definition 19. To establish $FA\lfloor CFA\rfloor \in \mathcal{P}_{\mathcal{PT}}$, we check that the defining process

$$vid, ca, rq.\overline{CFA}\langle id\rangle.\overline{id}\langle ca\rangle.\overline{id}\langle rq\rangle.(ca(rqnl).REQ\lfloor id, rqnl\rfloor + rq(nf).\overline{nf}\langle id\rangle.LD\lfloor id, nf\rfloor)$$

is in $\mathcal{P}_{\mathcal{PT}}$. The only free name is CFA. The send action $\overline{CFA}\langle id\rangle$ is the registration at the handler, denoted by $\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle$ in Definition 19. Then the free agent becomes an SQ process as it avoids the parallel composition. ◇

A *handler* process is a parallel composition of *connector processes* CN. To enable participants to register, a connector process contains receiving prefixes $\pi^R = p(x)$. Following the explanation above, a CN process should receive messages from finitely many participants. Therefore, we exclude recursive calls between CN processes, i.e., there is no process identifier K_C defined by a CN process (cf. process identifiers K_S in Definition 19). To make sure a CN process communicates with registered participants only, we include the side condition that the free names in a CN process are in $\mathcal{N}_{\mathcal{P}}$. This implies that all names—except those for registration purposes—are bound.

Definition 20 (Handler) The class of *handler* processes $\mathcal{P}_{\mathcal{HD}}$ with $HD \in \mathcal{P}_{\mathcal{HD}}$ is defined inductively in two steps:

$$CN ::= \Sigma_{i\in I}\pi_i^R.CN_i \mid \nu a.CN \mid K_H\lfloor\tilde{p}\rfloor$$
$$HD ::= CN_{\mathcal{N}_{\mathcal{P}}} \mid K_H\lfloor\tilde{p}\rfloor \mid HD_1 \mid HD_2,$$

where $CN_{\mathcal{N}_{\mathcal{P}}}$ is a connector process CN, which satisfies $fn\left(CN_{\mathcal{N}_{\mathcal{P}}}\right) \subseteq \mathcal{N}_{\mathcal{P}}$. A process identifier K_H is defined by a handler process $HD \in \mathcal{P}_{\mathcal{HD}}$.

In the car platoon system, the $MERGE$ process is in the class $\mathcal{P}_{\mathcal{HD}}$. ◇

Example 12 (Handler) Let again $\mathcal{N}_{\mathcal{P}} = \{CFA\}$. To see that $MERGE\lfloor CFA\rfloor$ is a handler process of the form $K_H\lfloor\tilde{p}\rfloor$, consider the process defining $MERGE$:

$$CFA(id_x).id_x(ca_x).id_x(rq_x).CFA(id_y).id_y(ca_y).id_y(rq_y).\overline{ca_y}\langle rq_x\rangle.MERGE\lfloor CFA\rfloor.$$

It is a CN process that receives twice on the distinguished name CFA. All other names are bound, i.e., the side condition that the free names are in $\mathcal{N}_{\mathcal{P}}$ holds. Hence, $MERGE\lfloor CFA\rfloor$ is a process in $\mathcal{P}_{\mathcal{HD}}$. ◇

Finite handler processes, denoted by FH, are built from participants and handler processes using choice composition, $\Sigma_{i\in I}\pi_i.FH_i$, and parallel composition, $FH_1 \mid FH_2$. Restricted names occur in participants and in handler processes only. Therefore, their use is well-controlled as a participant sends restricted names to a handler who distributes them among the other participants that are registered.

Definition 21 (Finite handler processes) The class of *finite handler processes* $\mathcal{P}_{\mathcal{FH}}$ with elements $FH \in \mathcal{P}_{\mathcal{FH}}$ is defined by

$$FH ::= PT \mid HD \mid \Sigma_{i\in I}\pi_i.FH_i \mid FH_1 \mid FH_2 \mid K_F\lfloor\tilde{a} \cup \tilde{p}\rfloor,$$

where $PT \in \mathcal{P}_{\mathcal{PT}}$, $HD \in \mathcal{P}_{\mathcal{HD}}$, and K_F is defined by $FH \in \mathcal{P}_{\mathcal{FH}}$. ◇

Example 13 concludes the explanation that the car platoon system is a finite handler process. The case study shows that finite handler processes may have an infinite number of reachable processes. Furthermore, there is no bound on the number of restricted names in all reachable processes.

Example 13 (Car platoon as finite handler process) Let $\mathcal{N}_{\mathcal{P}} = \{CFA\}$. We already observed that $FA\lfloor CFA\rfloor \in \mathcal{P}_{\mathcal{PT}}$ and $MERGE\lfloor CFA\rfloor \in \mathcal{P}_{\mathcal{HD}}$. The process $CREATE\lfloor CFA\rfloor$ is of the form $K_F\lfloor\tilde{a} \cup \tilde{p}\rfloor \in \mathcal{P}_{\mathcal{FH}}$ since it is defined by the finite handler process $FA\lfloor CFA\rfloor \mid CREATE\lfloor CFA\rfloor$. ◇

The main theorem in this section states that finite handler processes are structurally stationary.

Theorem 4 *Let $FH \in \mathcal{P}_{\mathcal{FH}}$, then FH is structurally stationary.*

In the remainder of the section we prove this result. We argue that all fragments reachable from a finite handler process consist of a bounded number of sequential processes. The maximal nesting of prefixes that receive on distinguished channels $\#_R(P)$ serves as the bound. For example, the function yields $\#_R(p(x).a(y).p(z) + \overline{p}\langle a\rangle) = \max\{2, 0\} = 2$.

Definition 22 For every process $P \in \mathcal{P}$, the *maximal nesting of prefixes that receive on distinguished channels*, i.e., prefixes $p(x)$, is given by $\#_R(P)$:

$$\begin{aligned}
\#_R(\pi^{RS}) &:= 1, \text{ if } \pi^{RS} = p(x), 0 \text{ otherwise}\\
\#_R(K\lfloor\tilde{a}\rfloor) &:= 0\\
\#_R(\Sigma_{i\in I}\pi_i^{RS}.P_i) &:= \max\{\#_R(\pi_i^{RS}) + \#_R(P_i) \mid i \in I\}\\
\#_R(P_1 \mid P_2) &:= \max\{\#_R(P_1), \#_R(P_2)\}\\
\#_R(\nu a.P) &:= \#_R(P).
\end{aligned}$$

Let P rely on the n defining equations $K_i(\tilde{x_i}) := P_i$. Taking them into account, we define $\max_R(P) := \max\{\#_R(P), \#_R(P_1), \ldots, \#_R(P_n)\}$. ◇

To prove the indicated boundedness requires a deeper understanding of the behaviour of finite handler processes. An induction on the reaction sequences reveals that every fragment reachable from $FH \in \mathcal{P}_{\mathcal{FH}}$ satisfies the constraints given by Definition 23. If the fragment consists of a single process, i.e., $number_{|}(F) = 1$, it is a process identifier, a connector process CN waiting for participants to register, or a participant $\nu\tilde{a}.(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i)$ ready to register at a connector. We have an SQ process in case a participant registered at a connector, was part of a fragment, but was separated from that fragment because restricted names were

forgotten. A finite handler process $\Sigma_{i\in I}\pi_i.FH_i$ communicating on public channels is also a correct fragment.

If the reachable fragment F contains more than one process, it is a parallel composition of SQ processes with at most one CN process, i.e., $F \equiv \nu\tilde{a}.(SQ_1 \mid \ldots \mid SQ_n \mid CN)$ with CN optional. The number of processes in this fragment, $number_{|}(F)$, satisfies $number_{|}(F) + \#_R(F) \leq \max_R(FH) + 1$. This inequality is crucial in the proof of Theorem 4. We explain it after the definition.

Definition 23 (Finite handler form) A fragment $F \in \mathcal{P}_{\mathcal{F}}$ reachable from $FH \in \mathcal{P}_{\mathcal{FH}}$ is in *finite handler form (fhf)*, if

1. either $number_{|}(F) > 1$ and $F \equiv \nu\tilde{a}.(SQ_1 \mid \ldots \mid SQ_n \mid CN)$ with CN optional, $fn(F) \subseteq \mathcal{N}_{\mathcal{P}}$, and
$$number_{|}(F) + \#_R(F) \leq \max_R(FH) + 1,$$
2. or F is of the form
$$F ::= K_P\lfloor\tilde{p}\rfloor \mid K_H\lfloor\tilde{p}\rfloor \mid K_F\lfloor\tilde{a}\cup\tilde{p}\rfloor \mid CN \mid \nu\tilde{a}.\left(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i\right) \mid SQ \mid \Sigma_{i\in I}\pi_i.FH_i$$
with $fn(CN) \subseteq \mathcal{N}_{\mathcal{P}}$, $fn\left(\nu\tilde{a}.\left(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i\right)\right) \subseteq \mathcal{N}_{\mathcal{P}}$, $fn(SQ) = \emptyset$, and $\#_R(CN) \leq \max_R(FH)$. ◇

Consider fragment $F \equiv \nu\tilde{a}.(SQ_1 \mid \ldots \mid SQ_n \mid CN)$ created by connector CN. Since neither the connector CN nor registered participants SQ_i contain parallel compositions, $number_{|}(F)$ does not increase by internal communication. Registered participants and connector communicate on restricted names. The free names of the fragment are the input prefixes of CN, which are in $\mathcal{N}_{\mathcal{P}}$. As input prefixes do not match, the fragment does not merge with a fragment or a connector. Sequential processes $\Sigma_{i\in I}\pi_i.FH_i$ do not use names in $\mathcal{N}_{\mathcal{P}}$. Hence, the only way to increase the number of processes in F is a communication between CN and a participant $\nu\tilde{a}.(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i)$.

The connector receives finitely many participants he may add to F. Every such communication decreases $\#_R(CN) = \#_R(F)$ by one and increases $number_{|}(F)$ by at most one. Initially, $\#_R(CN) \leq \max_R(FH)$ and all fragments consist of one process, i.e., $number_{|}(F) = 1$. We conclude $\#_R(F) + number_{|}(F) \leq \max_R(FH) + 1$.

Lemma 13 *Every fragment reachable from a finite handler process is in finite handler form, i.e.,*
$$\forall FH \in \mathcal{P}_{\mathcal{FH}} : \forall Q \in Reach(FH) : \forall F \in Frag(\nu(Q)) : F \text{ is in fhf}.$$

Before we continue with the proof of Lemma 13, we prove Theorem 4 as a corollary of the lemma and Theorem 2. Consider $FH \in \mathcal{P}_{\mathcal{FH}}$. With Lemma 13 we get the inequality $number_{|}(F) + \#_R(F) \leq \max_R(FH) + 1$ for all fragments F in all reachable processes Q. Thus $number_{|}(F) \leq \max_R(FH) + 1$. This means, the number of processes in all reachable fragments is bounded. Theorem 2 yields that FH is structurally stationary.

To establish Lemma 13, we do an induction on the length of the reaction sequence leading to the reachable process Q. We have to show that the fragments in $Frag(\nu(Q))$ are in *fhf*. The following Lemma 14 handles the base case, i.e., $Q_0 = FH$. It is also used in the induction step when a process identifier is called or processes in $\mathcal{P}_{\mathcal{FH}}$ communicate.

Lemma 14 *Let $PT \in \mathcal{P}_{\mathcal{PT}}$, $HD \in \mathcal{P}_{\mathcal{HD}}$, $FH \in \mathcal{P}_{\mathcal{FH}}$. If $F \in Frag(\nu(PT))$ or $F \in Frag(\nu(HD))$ or $F \in Frag(\nu(FH))$, then F is in fhf.*

Lemma 15 handles the induction step in the proof of Lemma 13 in case a participant registers at a connector process or a fragment consisting of several processes performs a reaction.

Lemma 15 *Let* $FH \in \mathcal{P}_{\mathcal{FH}}$ *with CN,* $\nu\tilde{a}.(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i)$*, and* F *with* $number_{|}(F) > 1$ *reachable fragments in fhf. Consider the reactions* $F \to R$*,* $CN \mid \nu\tilde{a}.(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i) \to R$*, and* $F \mid \nu\tilde{a}.(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i) \to R$*. In either case,* $R \equiv \Pi_{j\in J}F_j$ *so that all* F_j *are in fhf.*

In the proof of Lemma 13, the induction hypothesis assumes that all fragments in $Frag(\nu(Q_n))$ are in *fhf*. We distinguish all possible fragments in Definition 23 and all reactions $Q_n \to Q_{n+1}$. Lemmas 14 and 15 show that the resulting fragments in $Frag(\nu(Q_{n+1}))$ are in *fhf*.

The car platoon case study indicates that finite handler processes naturally arise when modelling systems with central control units. Theorem 4 ensures the property of structural stationarity holds for this system class. We also observed in Sect. 5 that various and important subclasses of the π-Calculus known from the literature are structurally stationary. This implies that the constraints in Theorems 2 and 3 are frequently satisfied. Structural stationarity is a common property of π-Calculus processes. We continue the discussion of related work in the following Sect. 7.

7 Related work

We first review the operational semantics of the π-Calculus. Then we summarise the outcomes of research on automatic verification tools for this calculus. To conclude, we discuss the relationship with work on structural congruence relations, which led to normal forms related to ours.

Operational Semantics To begin with, we discuss automata-theoretic semantics reflecting process behaviour. All of them explicitly represent the concurrency of sequential processes in the following sense. A state (place) of the associated automaton (or Petri net, respectively) represents a derivative of a sequential process [1,3,11,17,34]. Thus, processes are split along the parallel composition operator in contrast with our semantics decomposing along fragments, i.e., along substructures induced by the scopes of restricted names. Subsequently, we discuss a semantics that also represents structural information [33].

A Petri net semantics for the *small* π-Calculus is defined in [17]. The small π-Calculus does not contain choice composition and uses replication instead of recursion. The proposed semantics reflects the reaction relation. Name creation is handled using fresh global names, bound names in input prefixes are replaced by de Bruijn indices and replication by countably infinite union. In subsequent papers [13,15], Engelfriet and Gelsema show that the discriminating power of their semantics corresponds to extended and decidable versions of structural congruence. Due to their focus on structural congruence, the authors are not concerned with finiteness of the resulting Petri nets. In fact, the semantics immediately yields (1) infinite nets with (2) arcs that have countably infinite weights (denoted by ω), and (3) infinite markings (ω tokens are allowed), which makes it unusable for automatic verification of the system behaviour.

A translation of the π-Calculus into potentially infinite place/transition Petri nets with inhibitor arcs is presented in [3]. It models the *early* transition relation. Inhibitor arcs are employed in two ways. They check for the presence of restrictions on names but also allow for replacing choice with parallel composition. In a recent paper, the authors show that for

so-called *finite net processes* the resulting inhibitor nets are finite and primitive [4]. Primitive inhibitor nets enjoy the property that an inhibiting place can never be emptied, if a marking exceeds a certain bound. A result by Busi [5] shows that important properties of finite primitive inhibitor nets—including model checking of linear-time μ-Calculus—are decidable. The crucial characteristics of finite net processes is that they generate a bounded number of restricted names. So they are (1) bounded in depth but (2) incomparable with structurally stationary processes.

The main contribution in [4] is the study of non-interleaving and causal semantics for the π-Calculus. The authors argue that it is impossible to give a semantics in terms of standard place/transition Petri nets, which reflects the intended causality of processes. As our structural semantics reflects the interleaving behaviour of processes, it is coarser than the mentioned causal semantics and so their impossibility result does not apply.

The control reachability problem (CRP) asks for the reachability of a process containing a given process identifier [1]. The authors prove decidability for input bounded systems with unique receiver. A system is input bounded if the continuation of a name generating process is uniquely determined. The unique receiver condition demands that only the creating process listens on a restricted name. Combined, the conditions give an upper bound on the restricted names actually used for communication. Dead restricted names are replaced by generic ones. Decidability is obtained via a reduction to the coverability problem of Petri nets with transfer. We observe that CRP is decidable for structurally stationary processes. The relationship with input bounded unique receiver systems is a point for future research.

In [11,12] a translation into finite high-level Petri nets with read arcs is presented. The transitions of the net reflect strongly bisimilar the *indexed* transition system of the π-Calculus. A process is first translated into a context-based representation which removes restrictions. From this representation the high-level nets are obtained compositionally, i.e., for all remaining operators there is a corresponding net operator. Different instances of recursive processes are distinguished via invocation trails. Invocation itself is handled via marking equivalence. The authors aim at using the translation for automatic verification and report on successful first experiments [26].

History-Dependent automata (HD-automata) [32,34,36] handle names explicitly by associating to each state a set of names. Functions on transitions relate the names in label, source, and target states. HD-automata come equipped with bisimulation theory and techniques for computing irredundant and minimal representations. The *ground* and *early* π-Calculus semantics are translated into HD-automata. Depending on the translation, bisimilarity on the automata coincides with the corresponding bisimilarity on processes. Furthermore, bisimilarity on the automata coincides with isomorphism of the minimal representations. Finite HD-automata are gained for *finitary processes*. This covers the well-known *finite control processes* [10], where parallel composition is not used within recursions. In Sect. 5.2 we observed that finitary processes are structurally stationary.

The only semantics reflecting a notion of structure as discussed in Sect. 3.1 is the graph rewriting semantics in [33]. A hypergraph is constructed by mapping names to nodes and sequential processes to hyperedges. An arc indicates that a name is free in a process. The *early* transition relation of the π-Calculus is mimicked via a finite number of graph rewriting rules. The authors investigate observational equivalences based on the semantics.

π-Calculus Verification An extension of the modal μ-Calculus to cope with name creation and passing is proposed in [10]. The author presents a sound and complete proof system and a tableau-based model checking algorithm for finite control processes. The algorithm is

integrated in the MOBILITY WORKBENCH—initially designed to decide the *open* bisimilarities [40] between finite control processes [41].

The Spatial Logic [7] specifies structural as well as behavioural properties of processes. It contains operators to refer to subprocesses, environments, and the freshness of names. A model checking algorithm is available [6]. It is complete for the class of *bounded processes* where finitely many processes are reachable (from the process and all subprocesses). Bounded processes are structurally stationary by definition as finitely many reachable processes imply a finite number of reachable fragments.

Based on the theory of HD-automata, model and bisimulation checking tools for the π-Calculus are implemented [22]. Finite HD-automata are translated into ordinary finite automata, which makes finite state verification tools applicable. The π-logic, defined for model checking, is capable of referring to the identities of names. *Strong early* bisimilarity checking is performed by checking bisimilarity of the finite automata.

It does not seem likely that these decidability results for bisimilarity and branching-time logics carry over to structurally stationary processes because of the negative results for Petri nets [20,24].

Normal Forms Engelfriet and Gelsema use normal forms similar to ours but for the *small* π-Calculus in the proofs of their decidability and correspondence results [13,15]. In the *subconnected normal form* replications are moved inwards. Like our restricted form, the *cell normal form* strives for innermost restrictions, but also under prefixes. Our work differs in the way the normal form is exploited. While Engelfriet and Gelsema use it as technical tool, we use fragments as places in our semantics, i.e., we exploit the structure to finitely represent the process behaviour.

Recently, the authors investigated the structural congruence in [30] but for the small π-Calculus. For the class of *replication restricted processes* they prove decidability via a reduction to linear equations with natural coefficients [14]. It is based on a normal form that finds the connected subprocesses. These *webs* differ from our fragments in that they have outermost restrictions. In [16] the authors show decidability for restriction-free processes by a translation to reversible Petri nets. The transitions of the net model applications of the replication law $!P \equiv P \mid !P$. This semantics does not reflect any behavioural relation of processes.

8 Conclusion and future work

We presented a Petri net semantics that reflects the reaction relation of the monadic π-Calculus with recursion. The semantics known from the literature decompose processes along the parallel composition operator, i.e., they represent the sequential processes by places in the Petri net. We transform the process into a normal form we call *restricted form*. A process in restricted form is a parallel composition of *fragments*, groups of processes connected by restricted names. We decompose the process along the parallel composition of fragments, i.e., the fragments are the places in our Petri net semantics. The benefit is a finite Petri net representation for processes with unboundedly many restricted names and unbounded parallelism. Since fragments only depend on the restricted names that sequential processes share, we note that the extension of the restricted form and consequently of the semantics to the polyadic π-Calculus is straightforward.

Motivated by the idea of automatic verification, we prove exact semantical conditions for the finiteness of the semantics. In particular, we discover the novel characteristic functions of

depth and *breadth*. They indicate the quality of the connections induced by restricted names. A new syntactic class of processes, called *finite handler processes*, is defined that satisfies the finiteness conditions. By modelling a car platoon system in this class we demonstrate its usefulness. We apply our semantics to verify properties of different kinds for this case study using efficient algorithms.

To argue why our semantics is useful for verification, we discuss the aspects we believe constitute a useful semantics: retrievability, finiteness, expressiveness, and analysability. For the verification results to carry over from Petri nets to processes, the Petri net and the process under consideration should be closely related, i.e., retrievability should hold. We show that the transition systems are isomorphic in Theorem 1.

We are interested in exact analyses, which forbids domains that are Turing complete. We use Petri nets as domains. Infinite Petri nets are Turing complete, while finite ones are not. Thus, the finiteness of the semantic image is important. We show in Lemma 10 that exactly the structurally stationary processes have a finite Petri net representation. To apply our semantics for the verification of a wide range of systems, the class of structurally stationary processes should be expressive. We show that our class is powerful in Sect. 5.2. We provide two theorems that characterise structural stationarity. Applying these theorems immediately yields that the well-known *finite control processes* [10], *finitary processes* [32,34,36], and *restriction-free processes* [1] are structurally stationary. Furthermore, we define the class of *finite handler processes* with the aim of modelling client-server systems in Sect. 6. The characterisations show that finite handler processes are structurally stationary, Theorem 4.

A semantics useful for verification should produce nets where required properties can be inferred efficiently. We demonstrate the analysability of our semantics in Sect. 4 where we prove properties of different kinds without state space computations. Moreover, we map closed processes to *communication-free* Petri nets where every transition has exactly one place in its preset, Lemma 8. The graph structure of these nets allows us to conclude about the behaviour [21]. In [31] we showed that the structural semantics of finite control processes can be model checked highly efficient with unfolding-based verification tools.

We implemented the semantics in our tool PETRUCHIO [38,39]. The implementation relies on the coverability graph to detect the places that are markable simultaneously. As the size of the coverability graph is not bounded by a primitive recursive function in the size of the net, we plan to use more compact structures like unfoldings to decide simultaneous reachability. Unfoldings require the nets to be bounded, so to use them in the compilation requires the structural semantics as well as all intermediate nets to be bounded. Furthermore, algorithms are needed to update the unfoldings of intermediate nets to avoid recomputing them.

A different technique to compute the semantics is suggested by Theorem 2. If the bound k_s on the number of sequential processes in fragments is known, the set FG can be used as places in a Petri net $\mathcal{N}_{FG}[\![P]\!]$, which subsumes the structural semantics $\mathcal{N}[\![P]\!]$. To compute $\mathcal{N}_{FG}[\![P]\!]$ efficiently, precise approximations to the bound k_s and to the set of substitutions, which are applied to derivatives, need to be computed statically from the process P. In [2] a control flow analysis for the π-Calculus is proposed that overapproximates (1) for every channel the set of names that may be sent on it and (2) for every input prefix the channels which it may receive. It should be possible to adapt the technique to compute the required approximations.

The syntactic class of finite handler processes only covers a subset of the structurally stationary processes. For processes not contained it is unknown whether the computation of the semantics ever stops. We plan to include algorithms that approximate the answer to this question in our tool. In Sect. 5.2 we showed how the semantics of subprocesses may reveal unboundedness in breadth. Static analysis techniques may also be helpful.

The properties we verify in Sect. 4 are stated informally. In [7,10,22] temporal logics are proposed to formalise the correct behaviour of π-Calculus processes. These logics are able to specify the way names flow through a system. To handle such properties, we need to keep track of the identities of names. Following [32,34,36], transitions could be equipped with labels relating the names in pre- and postset. It deserves further investigation whether decidability results can be obtained for model checking linear-time variants of these logics.

Acknowledgments I am grateful to Ernst-Rüdiger Olderog and Maciej Koutny for helpful comments on preliminary versions of the paper and to Tim Strazny for implementing the translation. I also thank Roberto Gorrieri for discussions on the relationship with their semantics and one of the anonymous referees for suggesting to use the set *FG* in the computation of the semantics.

Appendix A: Proofs of Section 2

Proof of Lemma 1 The base cases are $K\lfloor\tilde{a}\rfloor$, $\mathbf{0}$, and the non-empty sum:

$$\begin{aligned}
&\mathcal{S}((\Sigma_{i\in I\neq\emptyset}\pi_i.P_i)\sigma)\\
(\text{Applic. }\sigma) &= \mathcal{S}(\Sigma_{i\in I\neq\emptyset}\pi_i\sigma.P_i\sigma)\\
(\text{Def. }\mathcal{S}) &= \{\Sigma_{i\in I\neq\emptyset}\pi_i\sigma.P_i\sigma\}\\
(\text{Applic. }\sigma) &= \{(\Sigma_{i\in I\neq\emptyset}\pi_i.P_i)\sigma\}\\
(\text{Def. }\mathcal{S}) &= \mathcal{S}(\Sigma_{i\in I\neq\emptyset}\pi_i.P_i)\sigma.
\end{aligned}$$

For the induction step, let $\mathcal{S}(P\sigma) = \mathcal{S}(P)\sigma$ and $\mathcal{S}(Q\sigma) = \mathcal{S}(Q)\sigma$. We first consider the restriction $\nu a.P$:

$$\begin{aligned}
&\mathcal{S}((\nu a.P)\sigma)\\
(\text{Applic. }\sigma) &= \mathcal{S}(\nu a.(P\sigma))\\
(\text{Def. }\mathcal{S}) &= \mathcal{S}(P\sigma)\\
(\text{Hypothesis}) &= \mathcal{S}(P)\sigma\\
(\text{Def. }\mathcal{S}) &= \mathcal{S}(\nu a.P)\sigma.
\end{aligned}$$

We then prove the equality for the parallel composition $P \mid Q$:

$$\begin{aligned}
&\mathcal{S}((P \mid Q)\sigma)\\
(\text{Applic. }\sigma) &= \mathcal{S}(P\sigma \mid Q\sigma)\\
(\text{Def. }\mathcal{S}) &= \mathcal{S}(P\sigma) \cup \mathcal{S}(Q\sigma)\\
(\text{Hypothesis}) &= \mathcal{S}(P)\sigma \cup \mathcal{S}(Q)\sigma\\
(\text{Applic. }\sigma\text{ to sets}) &= (\mathcal{S}(P) \cup \mathcal{S}(Q))\sigma\\
(\text{Def. }\mathcal{S}) &= \mathcal{S}(P \mid Q)\sigma.
\end{aligned}$$

This completes the proof. □

Proof of Lemma 2 With $P = K\lfloor\tilde{a}\rfloor$ we have $K\lfloor\tilde{a}\rfloor \in \mathcal{S}(P)$. The case of non-empty sums is analogous. The empty sum $\mathbf{0}$ is represented by $\Pi_{i\in\emptyset}P_i$.

For the induction step, let $P \equiv \nu\tilde{a}.(\Pi_{i\in I} P_i)$ with $P_i \in \mathcal{S}(P)$, $\tilde{a} \subseteq bn(P)$ and $Q \equiv \nu\tilde{c}.(\Pi_{j\in J} Q_j)$ with $Q_j \in \mathcal{S}(Q)$, $\tilde{c} \subseteq bn(Q)$. We consider $P \mid Q$:

$$\begin{aligned} & P \mid Q \\ (\text{Hypothesis}) &\equiv \nu\tilde{a}.(\Pi_{i\in I} P_i) \mid \nu\tilde{c}.(\Pi_{j\in J} Q_j) \\ (\text{Scope extr.}) &\equiv \nu\tilde{a}, \tilde{c}.(\Pi_{i\in I} P_i \mid \Pi_{j\in J} Q_j). \end{aligned}$$

The scope extrusions are correct since $\tilde{a} \cap \tilde{c} = \emptyset$, $\tilde{a} \cap fn(Q) = \emptyset$, and $\tilde{c} \cap fn(P) = \emptyset$. For the disjointness, we argue as follows. As we assume that a name is bound at most once, we have $bn(P) \cap bn(Q) = \emptyset$ in $P \mid Q$. Since $\tilde{a} \subseteq bn(P)$ and $\tilde{c} \subseteq bn(Q)$, the disjointness of $\tilde{a}$ and $\tilde{c}$ follows. Similarly, $bn(P \mid Q) \cap fn(P \mid Q) = \emptyset$ implies $\tilde{a} \cap fn(Q) = \emptyset$, since $\tilde{a} \subseteq bn(P) \subseteq bn(P \mid Q)$ and $fn(Q) \subseteq fn(P \mid Q)$. The case $\tilde{c} \cap fn(P) = \emptyset$ is similar.

The argumentation showed $\tilde{a} \cup \tilde{c} \subseteq bn(P \mid Q)$. To prove that the P_i and Q_j are in $\mathcal{S}(P \mid Q)$, we observe $P_i \in \mathcal{S}(P) \subseteq \mathcal{S}(P) \cup \mathcal{S}(Q) = \mathcal{S}(P \mid Q)$. Similarly for Q_j.

Consider the case $\nu a.P$. By the hypothesis, $\nu a.P$ is structurally congruent with $\nu a.\nu\tilde{a}.(\Pi_{i\in I} P_i)$, which is of the desired form. Also by the hypothesis $\tilde{a} \subseteq bn(P)$, which implies $\{a\} \cup \tilde{a} \subseteq \{a\} \cup bn(P) = bn(\nu a.P)$. With $P_i \in \mathcal{S}(P) = \mathcal{S}(\nu a.P)$ we conclude the proof. □

Appendix B: Proofs of Section 3

Proof of Lemma 4 By an induction on the structure of P we verify that $\nu(P)$ is in restricted form and structurally congruent with P. Similarly we prove that $\nu(-)$ does not change P_ν.

For $\mathbf{0} = \Sigma_{i\in\emptyset}\pi_i.P_i$ we have $\nu(\mathbf{0}) = \mathbf{0} = \Pi_{i\in\emptyset} F_i \in \mathcal{P}_\nu$. For a non-empty sum, the function $\nu(-)$ yields $\nu\left(\Sigma_{i\in I\neq\emptyset}\pi_i.P_i\right) = \Sigma_{i\in I\neq\emptyset}\pi_i.P_i$, which is a fragment in $\mathcal{P}_\mathcal{F} \subseteq \mathcal{P}_\nu$. Similarly for $K\lfloor\tilde{a}\rfloor$.

Assume $\nu(P_1), \nu(P_2) \in \mathcal{P}_\nu$ with $\nu(P_1) \equiv P_1$ and $\nu(P_2) \equiv P_2$. That $\nu(P_1)$ and $\nu(P_2)$ are in restricted form means $\nu(P_1) = \Pi_{i\in I} F_i$ and $\nu(P_2) = \Pi_{j\in J} G_j$. We first consider the parallel composition $P_1 \mid P_2$ and show $\nu(P_1 \mid P_2) \in \mathcal{P}_\nu$ as follows:

$$\nu(P_1 \mid P_2) = \nu(P_1) \mid \nu(P_2) = \Pi_{i\in I} F_i \mid \Pi_{j\in J} G_j \in \mathcal{P}_\nu.$$

The first equation is the definition of $\nu(-)$, the second the assumption on the form of $\nu(P_1)$ and $\nu(P_2)$. Structural congruence, i.e., $P_1 \mid P_2 \equiv \nu(P_1 \mid P_2)$, follows with the hypothesis and the definition of $\nu(-)$:

$$P_1 \mid P_2 \equiv \nu(P_1) \mid \nu(P_2) = \nu(P_1 \mid P_2)\,.$$

In the case of restriction, $\nu(\nu a.P_1) = \nu a.(\Pi_{i\in I_a} F_i) \mid \Pi_{i\in I\setminus I_a} F_i$ with $i \in I_a \subseteq I$ iff $a \in fn(F_i)$. Since by the hypothesis all F_i are fragments, $\nu a.(\Pi_{i\in I_a} F_i)$ is a fragment and $\nu(\nu a.P_1)$ is a parallel composition of fragments, i.e., $\nu(\nu a.P_1) \in \mathcal{P}_\nu$. To show $\nu a.P_1 \equiv \nu(\nu a.P_1)$, we observe

$$\nu a.P_1 \equiv \nu a.\,(\nu(P_1)) = \nu a.\,(\Pi_{i\in I} F_i) \equiv \nu a.\,\left(\Pi_{i\in I_a} F_i\right) \mid \Pi_{i\in I\setminus I_a} F_i = \nu(\nu a.P_1)\,.$$

Structural congruence follows from the hypothesis. We then replace $\nu(P_1)$ by $\Pi_{i\in I} F_i$ and use the scope extrusion rule. The last equation is the definition of $\nu(-)$.

We need to show that $\nu(-)$ does not change processes P_ν in restricted form. We show that it does not change fragments, i.e., $\nu(F) = F$. If we then have $P_\nu = \Pi_{i\in I} F_i$ we get $\nu(P_\nu) = \nu(\Pi_{i\in I} F_i) = \Pi_{i\in I}\nu(F_i) = \Pi_{i\in I} F_i = P_\nu$.

The base case where fragments are sequential processes is trivial. Assume $\nu(F_i) = F_i$ holds for $1 \leq i \leq n$. Consider $F = \nu a.(\Pi_{i=1}^n F_i)$. By definition, $\nu(F)$ computes the restricted form of $\Pi_{i=1}^n F_i$, $\nu\left(\Pi_{i=1}^n F_i\right) = \Pi_{i=1}^n \nu(F_i) = \Pi_{i=1}^n F_i$. The first equality is the definition of $\nu(-)$, the second holds by the hypothesis. We then determine $I_a \subseteq \{1, \ldots, n\}$ with $i \in I_a$ iff $a \in fn(F_i)$. Since $\nu a.(\Pi_{i=1}^n F_i)$ is a fragment, $a \in fn(F_i)$ for all i. Thus, $I_a = \{1, \ldots, n\}$ and $\nu(F) = \nu a.(\Pi_{i=1}^n F_i) = F$. □

Proof of Proposition 1 To show the implication from right to left we observe that all rules making up the equivalence $\widehat{\equiv}$ also hold for structural congruence. Thus, $\nu(P) \mathrel{\widehat{\equiv}} \nu(Q)$ implies $\nu(P) \equiv \nu(Q)$. Combined with $P \equiv \nu(P)$ from Lemma 4, we get $P \equiv Q$ with the transitivity of structural congruence. The proof of the reverse direction requires an induction on the derivations of structural congruence.

In the base case, we prove that the restricted equivalence holds for the axioms. As an example, we prove the cases of switching restricted names and using the rule for scope extrusion. Let $\nu(P) = \Pi_{i \in I} F_i$ and $i \in I_a \subseteq I$ iff $a \in fn(F_i)$. Let I_b be defined similarly. We distinguish between $I_a \cap I_b = \emptyset$ and $I_a \cap I_b \neq \emptyset$. In the first case:

$$\nu(\nu a.\nu b.P) = \nu a.\left(\Pi_{i \in I_a} F_i\right) \mid \nu b.\left(\Pi_{i \in I_b} F_i\right) \mid \Pi_{i \in I \setminus (I_a \cup I_b)} F_i \mathrel{\widehat{\equiv}} \nu(\nu b.\nu a.P)\,.$$

The equation holds because $I_a \cap I_b = \emptyset$. Consider $I_a \cap I_b \neq \emptyset$. By definition, $\nu(\nu b.P) = \nu b.(\Pi_{i \in I_b} F_i) \mid \Pi_{i \in I \setminus I_b} F_i$. Thus, the restricted form of $\nu a.\nu b.P$ is

$$\begin{aligned}\nu(\nu a.\nu b.P) &= \nu a.\left(\nu b.\left(\Pi_{i \in I_b} F_i\right) \mid \Pi_{i \in I_a \setminus I_b} F_i\right) \mid \Pi_{i \in I \setminus (I_a \cup I_b)} F_i \\ &\mathrel{\widehat{\equiv}} \nu b.\left(\nu a.\left(\Pi_{i \in I_a} F_i\right) \mid \Pi_{i \in I_b \setminus I_a} F_i\right) \mid \Pi_{i \in I \setminus (I_a \cup I_b)} F_i = \nu(\nu b.\nu a.P)\,.\end{aligned}$$

For $\widehat{\equiv}$ we argue as follows:

$$\begin{aligned}\nu a.\left(\nu b.\left(\Pi_{i \in I_b} F_i\right) \mid \Pi_{i \in I_a \setminus I_b} F_i\right) &\equiv \nu a.\nu b.\left(\Pi_{i \in I_a \cup I_b} F_i\right) \\ &\equiv \nu b.\left(\nu a.\left(\Pi_{i \in I_a} F_i\right) \mid \Pi_{i \in I_b \setminus I_a} F_i\right).\end{aligned}$$

Moreover, the first and last element are fragments, thus $\widehat{\equiv}$ holds.

We now prove $\nu(\nu a.(P \mid Q)) \mathrel{\widehat{\equiv}} \nu(P \mid \nu a.Q)$, if $a \notin fn(P)$. Let $\nu(P) = \Pi_{i \in I} F_i$ and $\nu(Q) = \Pi_{j \in J} G_j$, where $j \in J_a \subseteq J$ iff $a \in fn\left(G_j\right)$. Then

$$\begin{aligned}&\nu(\nu a.(P \mid Q)) \\ (a \notin fn(P) \text{ implies } a \notin fn(F_i)) &= \nu a.(\Pi_{j \in J_a} G_j) \mid \Pi_{i \in I} F_i \mid \Pi_{j \in J \setminus J_a} G_j \\ \text{(Ass. and Comm.)} &\mathrel{\widehat{\equiv}} \Pi_{i \in I} F_i \mid \nu a.(\Pi_{j \in J_a} G_j) \mid \Pi_{j \in J \setminus J_a} G_j \\ \text{(Def. } \nu(-)) &= \nu(P \mid \nu a.Q)\,.\end{aligned}$$

For the induction step, we assume that $P \equiv Q$ implies $\nu(P) \mathrel{\widehat{\equiv}} \nu(Q)$. We show that for all contexts where P and Q are placed in, i.e., for all $C[P] \equiv C[Q]$, it follows $\nu(C[P]) \mathrel{\widehat{\equiv}} \nu(C[Q])$. To begin with, we consider the choice composition:

$$\nu(\pi.P + M) = \pi.P + M \equiv \pi.Q + M = \nu(\pi.Q + M)\,.$$

Since $\pi.P + M$ and $\pi.Q + M$ are fragments we have $\nu(\pi.P + M) \mathrel{\widehat{\equiv}} \nu(\pi.Q + M)$.

For the parallel composition we have:

$$\nu(P \mid R) = \nu(P) \mid \nu(R) \mathrel{\widehat{\equiv}} \nu(Q) \mid \nu(R) = \nu(Q \mid R)\,.$$

To show $\widehat{\equiv}$ we argue as follows. Assume $\nu(P) = \Pi_{i \in I} F_i$ and $\nu(Q) = \Pi_{j \in J} G_j$. From $\nu(P) \mathrel{\widehat{\equiv}} \nu(Q)$ it follows that $|I| = |J| = n \in \mathbb{N}$. Wlog., let I and J be ordered so that

$F_i \equiv G_i$. Then

$$\begin{aligned}\nu(P) \mid \nu(R) &= F_1 \mid \Pi_{i=2}^n F_i \mid \nu(R)\\ &\mathrel{\widehat{\equiv}} G_1 \mid \Pi_{i=2}^n F_i \mid \nu(R)\\ &\mathrel{\widehat{\equiv}} F_2 \mid G_1 \mid \Pi_{i=3}^n F_i \mid \nu(R)\\ &\mathrel{\widehat{\equiv}} G_2 \mid G_1 \mid \Pi_{i=3}^n F_i \mid \nu(R) \mathrel{\widehat{\equiv}} \Pi_{j\in J} G_j \mid \nu(R) = \nu(Q) \mid \nu(R)\,.\end{aligned}$$

In the first equation, we replace the term $\nu(P)$ by $\Pi_{i\in I} F_i$ and drag F_1 out of the parallel composition. We use $F \mid P_\nu \mathrel{\widehat{\equiv}} G \mid P_\nu$ if $F \equiv G$ to replace F_1 by G_1. We now drag outside the following $n-1$ fragments in $\nu(P)$ and replace them by the fragments in $\nu(Q)$. We end up with the parallel composition of fragments in $\nu(Q)$, which we replace by the term $\nu(Q)$.

The final case is restriction. Assume again $\nu(P) = \Pi_{i\in I} F_i$ and $\nu(Q) = \Pi_{j\in J} G_j$. From $\mathrel{\widehat{\equiv}}$ we have $|I| = |J|$ and we assume both sets ordered so that $F_i \equiv G_i$. Free names are preserved by structural congruence, thus $a \in fn(F_i)$ if and only if $a \in fn(G_i)$. Let $I_a \subseteq I$ so that $i \in I_a$ iff $a \in fn(F_i)$. It follows:

$$\nu(\nu a.P) = \nu a.\left(\Pi_{i\in I_a} F_i\right) \mid \Pi_{i\in I\setminus I_a} F_i \mathrel{\widehat{\equiv}} \nu a.\left(\Pi_{i\in I_a} G_i\right) \mid \Pi_{i\in I\setminus I_a} G_i = \nu(\nu a.Q)\,.$$

Since $F_i \equiv G_i$ for all $i \in I_a$, we get $\nu a.(\Pi_{i\in I_a} F_i) \equiv \nu a.(\Pi_{i\in I_a} G_i)$. Both are fragments, so $\mathrel{\widehat{\equiv}}$ holds. The remaining F_i are replaced by structurally congruent G_i like in the case of parallel composition. □

Proof of Lemma 5 That $dec(P_\nu \mid Q_\nu) = dec(Q_\nu \mid P_\nu)$ and $dec(P_\nu \mid (Q_\nu \mid R_\nu)) = dec((P_\nu \mid Q_\nu) \mid R_\nu)$ is immediate. We consider the rule $F \mid P_\nu \mathrel{\widehat{\equiv}} G \mid P_\nu$ with $F \equiv G$. Let $P_\nu = \Pi_{i\in I} F_i$. For the class $[F]$, we get $(dec(F \mid \Pi_{i\in I} F_i))([F]) := 1 + |I_F| =: (dec(G \mid \Pi_{i\in I} F_i))([F])$, where $I_F \subseteq I$ with $i \in I_F$ iff $F \equiv F_i$. For $H \not\equiv F$, we have $H \not\equiv G$ and therefore $(dec(F \mid \Pi_{i\in I} F_i))([H]) := |I_H| =: (dec(G \mid \Pi_{i\in I} F_i))([H])$. Thus, $dec(F \mid \Pi_{i\in I} F_i) = dec(G \mid \Pi_{i\in I} F_i)$.

Assume $dec(P_\nu) = dec(Q_\nu)$ with $P_\nu = \Pi_{i\in I} F_i$ and $Q_\nu = \Pi_{j\in J} G_j$. Equality of the functions, $dec(P_\nu) = dec(Q_\nu)$, implies $supp(dec(P_\nu)) = supp(dec(Q_\nu))$. But $(dec(P_\nu))([F]) \neq 0$ if and only if $[F] \in Frag(P_\nu)/_\equiv$. Thus $supp(dec(P_\nu)) = Frag(P_\nu)/_\equiv$. We conclude $Frag(P_\nu)/_\equiv = Frag(Q_\nu)/_\equiv = \{[H_1], \ldots, [H_p]\}$ with representatives $H_1, \ldots, H_p$. We replace every fragment F_i by its representative $H_{\phi(i)}$, i.e., $\phi : I \to \{1, \ldots, p\}$ with $F_i \equiv H_{\phi(i)}$. We then reorder the fragments H. With $I_H \subseteq I$ defined by $i \in I_H$ iff $H \equiv F_i$, there are $|I_H|$ fragments F_i which are mapped to H. Let $J_H \subseteq J$ be defined similarly. We compute:

$$\begin{aligned}P_\nu &= \Pi_{i\in I} F_i\\ (H \text{ representative}) &\mathrel{\widehat{\equiv}} \Pi_{i\in I} H_{\phi(i)}\\ (\text{Reordering of fragments}) &\mathrel{\widehat{\equiv}} \Pi_{[H]\in Frag(P_\nu)/_\equiv} \Pi^{|I_H|} H\\ (\text{Def. } (dec(P_\nu))([H])) &= \Pi_{[H]\in Frag(P_\nu)/_\equiv} \Pi^{(dec(P_\nu))([H])} H\\ (\text{Assump. } dec(P_\nu) = dec(Q_\nu)) &= \Pi_{[H]\in Frag(P_\nu)/_\equiv} \Pi^{(dec(Q_\nu))([H])} H\\ (Frag(P_\nu)/_\equiv = Frag(Q_\nu)/_\equiv) &= \Pi_{[H]\in Frag(Q_\nu)/_\equiv} \Pi^{(dec(Q_\nu))([H])} H\\ (\text{Similarly for } Q_\nu) &\mathrel{\widehat{\equiv}} Q_\nu.\end{aligned}$$

This proves $P_\nu \mathrel{\widehat{\equiv}} Q_\nu$. □

Proof of Lemma 7 We show that finiteness of the set of places implies finiteness of the set of transitions. Let the places be $[F_1], \ldots, [F_n]$. For every F_i and every pair F_i, F_j there are

up to structural congruence finitely many Q with $F_i \to Q$ and $F_i \mid F_j \to Q$ because the reaction relation is image-finite up to structural congruence. Thus, there are finitely many combinations $([F_i], [Q])$ and $([F_i \mid F_j], [Q])$, respectively. The set of transitions is finite. □

Proof of Lemma 9

$$
\begin{aligned}
&\exists([F],[Q]) \in T : M_1(s) \geq \lambda(s,([F],[Q])), \text{ for all } s \in {}^\bullet([F],[Q]) \\
&\qquad \wedge M_2(s) = M_1(s) - \lambda(s,([F],[Q])) + \lambda((([F],[Q]),s), \text{ for all } s \in S \\
\Leftrightarrow\ &\text{(Definition of } \lambda) \\
&\exists([F],[Q]) \in T : M_1([F]) \geq \lambda([F],([F],[Q])) = 1 \\
&\qquad \wedge M_2(s) = M_1(s) - \lambda(s,([F],[Q])) + \lambda((([F],[Q]),s), \text{ for all } s \in S \\
\Leftrightarrow\ &\text{(Definition of } f, M_i = f([P_i]); S = \mathit{Frag}\,(\nu(\mathit{Reach}\,(P)))/_{\equiv}) \\
&\qquad \exists([F],[Q]) \in T : (\mathit{dec}\,(\nu(P_1)))\,([F]) \geq 1 \\
&\qquad\qquad \wedge (\mathit{dec}\,(\nu(P_2)))\,([F']) = (\mathit{dec}\,(\nu(P_1)))\,([F']) \\
&\qquad\qquad\qquad -\lambda([F'],([F],[Q])) + \lambda((([F],[Q]),[F']), \text{ for all } [F'] \in S \\
\Leftrightarrow\ &\text{(Definition of } \lambda) \\
&\qquad \exists([F],[Q]) \in T : (\mathit{dec}\,(\nu(P_1)))\,([F]) \geq 1 \\
&\qquad\qquad \wedge (\mathit{dec}\,(\nu(P_2)))\,([F']) = (\mathit{dec}\,(\nu(P_1)))\,([F']) \\
&\qquad\qquad\qquad - (\mathit{dec}\,(F))\,([F']) + (\mathit{dec}\,(\nu(Q)))\,([F']), \text{ for all } [F'] \in S \\
\Leftrightarrow\ &\text{(Definition of } (\mathit{dec}\,(\nu(P_1)))\,([F])) \\
&\qquad \exists([F],[Q]) \in T, P' \in \mathcal{P} : \nu(P_1) \equiv F \mid P' \\
&\qquad\qquad \wedge (\mathit{dec}\,(\nu(P_2)))\,([F']) = (\mathit{dec}\,(\nu(P_1)))\,([F']) \\
&\qquad\qquad\qquad - (\mathit{dec}\,(F))\,([F']) + (\mathit{dec}\,(\nu(Q)))\,([F']), \text{ for all } [F'] \in S \\
\Leftrightarrow\ &\text{(Definition of } T) \\
&\qquad \exists F \in \mathit{Frag}\,(\nu(\mathit{Reach}\,(P)))\,, Q, P' \in \mathcal{P} : F \to Q \wedge \nu(P_1) \equiv F \mid P' \\
&\qquad\qquad \wedge (\mathit{dec}\,(\nu(P_2)))\,([F']) = (\mathit{dec}\,(\nu(P_1)))\,([F']) \\
&\qquad\qquad\qquad - (\mathit{dec}\,(F))\,([F']) + (\mathit{dec}\,(\nu(Q)))\,([F']), \text{ for all } [F'] \in S.
\end{aligned}
$$

All fragments in F, $\nu(Q)$, $\nu(P_1)$, $\nu(P_2)$ are in S. Hence, for all fragments $[F'] \notin S$ we get $(\mathit{dec}\,(F))\,([F']) = (\mathit{dec}\,(\nu(Q)))\,([F']) = (\mathit{dec}\,(\nu(P_i)))\,([F']) = 0$. Thus, the equality $(\mathit{dec}\,(\nu(P_2)))\,([F']) = (\mathit{dec}\,(\nu(P_1)))\,([F']) - (\mathit{dec}\,(F))\,([F']) + (\mathit{dec}\,(\nu(Q)))\,([F'])$ holds for all $[F'] \in \mathcal{P}_{\mathcal{F}}/_{\equiv}$:

$$
\begin{aligned}
\Leftrightarrow\ &\exists F \in \mathit{Frag}\,(\nu(\mathit{Reach}\,(P)))\,, Q, P' \in \mathcal{P} : F \to Q \wedge \nu(P_1) \equiv F \mid P' \\
&\qquad \wedge \mathit{dec}\,(\nu(P_2)) = \mathit{dec}\,(\nu(P_1)) - \mathit{dec}\,(F) + \mathit{dec}\,(\nu(Q)) \\
\Leftrightarrow\ &\exists F \in \mathit{Frag}\,(\nu(\mathit{Reach}\,(P)))\,, Q, P' \in \mathcal{P} : F \to Q \\
&\qquad \wedge P_1 \equiv F \mid P' \wedge P_2 \equiv Q \mid P'.
\end{aligned}
$$

The last equivalence requires some consideration. With Lemma 4, $P_1 \equiv \nu(P_1)$. Thus, $\nu(P_1) \equiv F \mid P'$ iff $P_1 \equiv F \mid P'$. With Proposition 1, $P_1 \equiv F \mid P'$ implies $\nu(P_1) \mathrel{\widehat{\equiv}} \nu\big(F \mid P'\big) = F \mid \nu\big(P'\big)$. With Lemmas 5 and 6 we conclude $\mathit{dec}\,(\nu(P_1)) = \mathit{dec}\,\big(F \mid \nu\big(P'\big)\big) = \mathit{dec}\,(F) +$

$dec\left(\nu\left(P'\right)\right)$. Using this equality we have:

$$\begin{aligned} & & dec\left(\nu(P_2)\right) \\ & & = dec\left(\nu(P_1)\right) - dec\left(F\right) + dec\left(\nu(Q)\right) \\ \left(dec\left(\nu(P_1)\right) = dec\left(F\right) + dec\left(\nu\left(P'\right)\right)\right) & = & dec\left(\nu\left(P'\right)\right) + dec\left(\nu(Q)\right) \\ \text{(Lemma 6)} & = & dec\left(\nu\left(P' \mid Q\right)\right). \end{aligned}$$

Thus $dec\left(\nu\left(P' \mid Q\right)\right) = dec\left(\nu(P_2)\right)$. Lemma 5 applies and the equality is equivalent with $\nu(P_2) \mathrel{\widehat{\equiv}} \nu\left(P' \mid Q\right)$. With Proposition 1, this is equivalent with $P_2 \equiv Q \mid P'$. This shows the last equivalence. □

The following Lemma 16 is the analogue of Lemma 9 for the case that two fragments communicate. The proof is similar to that of Lemma 9.

Lemma 16 *Consider the function* $f : Reach\left(P\right)/_{\equiv} \to Reach\left(\mathcal{N}[\![P]\!]\right)$ *from Theorem 1. For* $M_1 = f([P_1])$ *and* $M_2 = f([P_2])$ *we have:*

$$\exists([F_1 \mid F_2], [Q]) \in T : M_1(s) \geq \lambda(s, ([F_1 \mid F_2], [Q])), \text{ for all } s \in {}^\bullet([F_1 \mid F_2], [Q])$$

$$\text{and } M_2(s) = M_1(s) - \lambda(s, ([F_1 \mid F_2], [Q])) + \lambda(([F_1 \mid F_2], [Q]), s), \text{ for all } s \in S$$

if and only if

$$\exists F_1, F_2 \in Frag\left(\nu(Reach\left(P\right))\right), Q, P' \in \mathcal{P} : F_1 \mid F_2 \to Q$$

$$\text{and } P_1 \equiv F_1 \mid F_2 \mid P' \text{ and } P_2 \equiv Q \mid P'.$$

Proof of Theorem 1

Initial States The initial states coincide by the definition of $\mathcal{N}$ and the definition of f, $M_0 := dec\left(\nu(P)\right) =: f([P])$.

Injectivity Let $[Q_1], [Q_2] \in Reach\left(P\right)/_{\equiv}$. With Proposition 1, $Q_1 \not\equiv Q_2$ implies $\nu(Q_1) \mathrel{\widehat{\not\equiv}} \nu(Q_2)$. With Lemma 5 we have $dec\left(\nu(Q_1)\right) \neq dec\left(\nu(Q_2)\right)$. Since $f([Q_1]) = dec\left(\nu(Q_1)\right) \neq dec\left(\nu(Q_2)\right) = f([Q_2])$, f is injective.

Surjectivity Let M be a reachable marking. We have to show that a process $[Q]$ is reachable with $f([Q]) = M$. We prove this by induction on the length of the transition sequences. In the base case, we consider the empty sequence, i.e., we reach M_0 in the Petri net. We already observed $M_0 = f([P])$.

Assume for M_n we have the reachable process $[P_n]$ with $f([P_n]) = M_n$. Consider M_{n+1} with $M_n[t_{n+1}\rangle M_{n+1}$. We prove the existence of $[P_{n+1}]$ with $P_n \to P_{n+1}$ and $f([P_{n+1}]) = M_{n+1}$. By definition of the transition relation for Petri nets we have

$$\begin{aligned} & M_n[t_{n+1}\rangle M_{n+1} \\ \Leftrightarrow\ & M_n([F]) \geq \lambda([F], t_{n+1}), \text{ for all } [F] \in {}^\bullet t_{n+1} \text{ and} \\ & M_{n+1}([F]) = M_n([F]) - \lambda([F], t_{n+1}) + \lambda(t_{n+1}, [F]), \text{ for all } [F] \in S. \end{aligned}$$

There are two kinds of transitions, $t_{n+1} = ([F], [Q])$ with $F \to Q$ and $t_{n+1} = ([F_1 \mid F_2], [Q])$ with $F_1 \mid F_2 \to Q$. We discuss the first case.

$$
\begin{aligned}
& M_n([F']) \geq \lambda([F'], ([F], [Q])), \text{ for all } [F'] \in {}^\bullet([F], [Q]) \\
\Rightarrow\ & M_n([F]) \geq \lambda([F], ([F], [Q])) = 1 \\
\Rightarrow\ & \text{(Hypothesis: } M_n = f([P_n]), \text{ definition of } f) \\
& (dec\,(\nu(P_n)))\,([F]) \geq 1 \\
\Rightarrow\ & \text{(Definition of } (dec\,(\nu(P_n)))\,([F])) \\
& \exists P' \in \mathcal{P} : \nu(P_n) \equiv F \mid P' \\
\Rightarrow\ & \text{(By definition of } t_{n+1} \text{ it holds } F \to Q, \text{ Rules (Par) and (Struct))} \\
& \exists P' \in \mathcal{P} : P_n \equiv \nu(P_n) \equiv F \mid P' \to Q \mid P'.
\end{aligned}
$$

This means $P_n \to Q \mid P'$. To prove $f([Q \mid P']) = M_{n+1}$, we observe the equality $dec\left(\nu\left(Q \mid P'\right)\right) = dec\left(\nu(Q) \mid \nu\left(P'\right)\right) = dec\,(\nu(Q)) + dec\left(\nu\left(P'\right)\right)$. Furthermore, from $P_n \equiv F \mid P'$ we derive $\nu(P_n) \mathrel{\widehat{\equiv}} \nu\left(F \mid P'\right) = F \mid \nu\left(P'\right)$ with Proposition 1. Lemmas 5 and 6 yield $dec\,(\nu(P_n)) = dec\left(F \mid \nu\left(P'\right)\right) = dec\,(F) + dec\left(\nu\left(P'\right)\right)$. Equivalently stated: $dec\left(\nu\left(P'\right)\right) = dec\,(\nu(P_n)) - dec\,(F)$. Since $Q \mid P'$ is reachable, $[F'] \notin S$ implies $\left(dec\left(\nu\left(Q \mid P'\right)\right)\right)([F']) = 0$. We compute for $[F'] \in S$:

$$
\begin{aligned}
\left(dec\left(\nu\left(Q \mid P'\right)\right)\right)([F']) &= (dec\,(\nu(Q)))\,([F']) + \left(dec\left(\nu\left(P'\right)\right)\right)([F']) \\
&= (dec\left(\nu\left(P'\right)\right) = dec\,(\nu(P_n)) - dec\,(F) \text{ explained above}) \\
&\quad (dec\,(\nu(Q)))\,([F']) + (dec\,(\nu(P_n)))\,([F']) - (dec\,(F))\,([F']) \\
&= (\text{Definition } \lambda(-,-);\ dec\,(\nu(P_n)) = f([P_n]) = M_n) \\
&\quad \lambda(([F], [Q]), [F']) + M_n([F']) - \lambda([F'], ([F], [Q])) \\
&= (\text{Equation in Def. } M_n[([F], [Q])\rangle M_{n+1}) \\
&\quad M_{n+1}([F']).
\end{aligned}
$$

This shows $f([Q \mid P']) = M_{n+1}$ in the case $t_{n+1} = ([F], [Q])$. The proof for $t_{n+1} = ([F_1 \mid F_2], [Q])$ is similar. We conclude surjectivity.

Strong Graph Homomorphism Let P_1 and P_2 be two reachable processes and $M_1 = f([P_1])$, $M_2 = f([P_2])$. We show $M_1 \to M_2$ if and only if $[P_1] \to [P_2]$. The proof amounts to applying Lemma 9 to switch from Petri net to process level. We then need the fact that at most two processes are involved in a reaction.

$$
\begin{aligned}
& M_1 \to M_2 \\
\Leftrightarrow\ & \text{(Definition of } \to) \\
& \exists t \in T : M_1[t\rangle M_2 \\
\Leftrightarrow\ & \text{(Definition of } [t\rangle) \\
& \exists t \in T : M_1(s) \geq \lambda(s, t), \text{ for all } s \in {}^\bullet t \\
& \qquad \wedge M_2(s) = M_1(s) - \lambda(s, t) + \lambda(t, s), \text{ for all } s \in S \\
\Leftrightarrow\ & \text{(Definition of } T) \\
& \exists([F], [Q]) \in T : M_1(s) \geq \lambda(s, ([F], [Q])), \text{ for all } s \in {}^\bullet([F], [Q]) \\
& \qquad \wedge M_2(s) = M_1(s) - \lambda(s, ([F], [Q])) + \lambda((([F], [Q]), s), \text{ for all } s \in S \\
& \vee\ \exists([F_1 \mid F_2], [Q]) \in T : M_1(s) \geq \lambda(s, ([F_1 \mid F_2], [Q])), \text{ for all } s \in {}^\bullet([F_1 \mid F_2], [Q])
\end{aligned}
$$

$$\wedge\, M_2(s) = M_1(s) - \lambda(s, ([F_1 \mid F_2], [Q])) + \lambda(([F_1 \mid F_2], [Q]), s), \text{ for all } s \in S$$

$\Leftrightarrow$ (Lemmas 9 and 16)

$$\exists F \in Frag\,(\nu(Reach\,(P)))\,,\, Q, P' \in \mathcal{P} : F \to Q$$
$$\wedge\, P_1 \equiv F \mid P' \wedge P_2 \equiv Q \mid P'$$
$$\vee\, \exists F_1, F_2 \in Frag\,(\nu(Reach\,(P)))\,,\, Q, P' \in \mathcal{P} : F_1 \mid F_2 \to Q$$
$$\wedge\, P_1 \equiv F_1 \mid F_2 \mid P' \wedge P_2 \equiv Q \mid P'.$$

The following implication from left to right holds with the Rules (Par) and (Struct). The reverse direction is a consequence of Lemma 1.2.20 in [40]. It reveals that at most two sequential processes are involved in a reaction. Since a sequential process is located in exactly one fragment, reactions require at most two fragments.

$$\begin{aligned} &\Leftrightarrow P_1 \to P_2 \\ &\Leftrightarrow \text{(Rule (Struct))} \\ &\quad [P_1] \to [P_2]. \end{aligned}$$

Retrievability We check $retrieve\,(f([Q])) = [Q]$. Let $Q \in Reach\,(P)$ be a reachable process. Let $\nu(Q) = \Pi_{i \in I} F_i$ with $Frag\,(\nu(Q))/_{\equiv} = \{[G_1], \ldots, [G_m]\}$. For every G let the index set $I_G \subseteq I$ be defined by $i \in I_G$ iff $F_i \equiv G$. We show that

$$\Pi_{[F] \in supp(dec(\nu(Q)))} \Pi^{(dec(\nu(Q)))([F])} F \equiv Q.$$

In the proof, $\phi : I \to \{1, \ldots, m\}$ maps the index of a fragment F_i to the index of the representative $G_{\phi(i)} \equiv F_i$:

$$\begin{aligned} Q & \\ \text{(Lemma 4)} \equiv\ & \nu(Q) \\ =\ & \Pi_{i \in I} F_i \\ \text{(G representative)} \equiv\ & \Pi_{i \in I} G_{\phi(i)} \\ \text{(Reordering of fragments)} \equiv\ & \Pi_{[G] \in Frag(\nu(Q))/_{\equiv}} \Pi^{|I_G|} G \\ \text{(Def. } dec\,(\nu(Q))) \equiv\ & \Pi_{[G] \in Frag(\nu(Q))/_{\equiv}} \Pi^{(dec(\nu(Q)))([G])} G \\ (Frag\,(\nu(Q))/_{\equiv} = supp\,(dec\,(\nu(Q)))) =\ & \Pi_{[G] \in supp(dec(\nu(Q)))} \Pi^{(dec(\nu(Q)))([G])} G. \end{aligned}$$

Thus, $retrieve\,(f([Q])) = [\Pi_{[G] \in supp(dec(\nu(Q)))} \Pi^{(dec(\nu(Q)))([G])} G] = [Q]$. □

Proof of Proposition 2 From $P \equiv Q$ we have $Reach\,(P) = Reach\,(Q)$ with Rule (Struct). Thus $Frag\,(\nu(Reach\,(P))) = Frag\,(\nu(Reach\,(Q)))$ and with this we conclude $Frag\,(\nu(Reach\,(P)))/_{\equiv} = Frag\,(\nu(Reach\,(Q)))/_{\equiv}$, i.e., $S_{\mathcal{N}[\![P]\!]} = S_{\mathcal{N}[\![Q]\!]}$. Since the transition sets as well as the weight functions depend on S only, we derive $T_{\mathcal{N}[\![P]\!]} = T_{\mathcal{N}[\![Q]\!]}$ and $\lambda_{\mathcal{N}[\![P]\!]} = \lambda_{\mathcal{N}[\![Q]\!]}$. With Proposition 1, $P \equiv Q$ implies $\nu(P) \mathrel{\widehat{\equiv}} \nu(Q)$. From this, we conclude $dec\,(\nu(P)) = dec\,(\nu(Q))$ with Lemma 5. Thus, the initial markings coincide and the nets are equal.

The other direction holds with Theorem 1. Let $M_{0,P}$=$M_{0,Q}$ be the initial markings in $\mathcal{N}[\![P]\!]$=$\mathcal{N}[\![Q]\!]$. We have $[P] = retrieve\,(M_{0,P}) = retrieve\,(M_{0,Q}) = [Q]$. □

Appendix C: Proofs of Section 5

Proof of Lemma 10 With an application of Lemma 7, the finiteness of $\mathcal{N}[\![P]\!]$ is equivalent to the finiteness of the set of places, $Frag\,(\nu(Reach\,(P)))/_{\equiv}$. The finiteness of $Frag\,(\nu(Reach\,(P)))/_{\equiv}$ is equivalent to structural stationarity. □

Proof of Proposition 3 We prove the claim for $Q \in Reach\,(P)$ via an induction on the length of the reaction sequences. We derive the statement for F as a corollary. The base case is the empty sequence, i.e., $Q_0 = P$. By Lemma 2, $P \equiv \nu\tilde{a}.(\Pi_{i\in I} P_i) = \nu\tilde{a}.(\Pi_{i\in I} P_i id)$ with $P_i \in \mathcal{S}(P) \subseteq der(P) \subseteq derivatives(P)$ and id the identity function. To show $id : fn\,(P_i) \to fn\,(P) \bigcup \tilde{a}$, consider a name $a \in fn\,(P_i)$, which is not in $\tilde{a}$. Then $a \in fn\,(\nu\tilde{a}.(\Pi_{i\in I} P_i)) = fn\,(P)$ as the free names are invariant under structural congruence.

In the induction step, let $Q_n \equiv \nu\tilde{a}.(\Pi_{i\in I} R_i\sigma_i)$ with $R_i \in derivatives(P)$ and $\sigma_i : fn\,(R_i) \to fn\,(P) \cup \tilde{a}$. We consider $Q_n \to Q_{n+1}$ via a communication between $R_1\sigma_1$ and $R_2\sigma_2$. The remaining cases, i.e., consumption of τ prefixes in a process $R_1\sigma_1 = \tau.R'_1\sigma_1 + M_1\sigma_1$ and calls to process identifiers in $R_1\sigma_1 = K\lfloor\tilde{x}\rfloor\sigma_1$, are similar. Let

$$R_1\sigma_1 = \overline{x_1\sigma_1}\langle y_1\sigma_1\rangle.R'_1\sigma_1 + M_1\sigma_1 = \overline{a}\langle b\rangle.R'_1\sigma_1 + M_1\sigma_1$$
$$R_2\sigma_2 = x_2\sigma_2(y_2).R'_2\sigma_2 + M_2\sigma_2 = a(y_2).R'_2\sigma_2 + M_2\sigma_2.$$

We have $R_1\sigma_1 \mid R_2\sigma_2 \to R'_1\sigma_1 \mid R'_2\sigma_2\{y_1\sigma_1/y_2\}$. Thus for Q_n we get

$$Q_n \to \nu\tilde{a}.(R'_1\sigma_1 \mid R'_2\sigma_2\{y_1\sigma_1/y_2\} \mid \Pi_{i\in I\setminus\{1,2\}} R_i\sigma_i) =: Q_{n+1}.$$

We transform Q_{n+1} into the required form. An application of Lemma 2 yields the congruence $R'_2\sigma_2\{y_1\sigma_1/y_2\} \equiv \nu\tilde{a}_2.(\Pi_{j\in J} R_{2,j})$ with $R_{2,j} \in \mathcal{S}(R'_2\sigma_2\{y_1\sigma_1/y_2\})$ and $\tilde{a}_2 \subseteq bn\left(R'_2\sigma_2\{y_1\sigma_1/y_2\}\right)$. We assume bound names to be disjoint with free names, so $\tilde{a}_2 \cap \left(fn\left(R'_1\sigma_1\right) \cup \bigcup_{i\in I\setminus\{1,2\}} fn\,(R_i\sigma_i)\right) = \emptyset$. Lemma 1 gives:

$$\begin{aligned} R_{2,j} &\in \mathcal{S}(R'_2\sigma_2\{y_1\sigma_1/y_2\}) \\ &= \mathcal{S}(R'_2\sigma_2)\{y_1\sigma_1/y_2\} \\ &= \mathcal{S}(R'_2)\sigma_2\{y_1\sigma_1/y_2\}. \end{aligned}$$

We observe $\mathcal{S}(R'_2) \subseteq der(R'_2) \subseteq der(R_2) \subseteq derivatives(P)$. The first two inclusions hold by the definitions of $\mathcal{S}$ and der, the last follows from the induction hypothesis $R_2 \in derivatives(P)$, which implies $der(R_2) \subseteq derivatives(P)$. With this argumentation, we have $R_{2,j} \in derivatives(P)\sigma_2\{y_1\sigma_1/y_2\}$, i.e., there is $P_{2,j} \in derivatives(P)$ with $R_{2,j} = P_{2,j}\sigma_2\{y_1\sigma_1/y_2\}$. To sum up,

$$\begin{aligned} Q_{n+1} &= \nu\tilde{a}.(R'_1\sigma_1 \mid R'_2\sigma_2\{y_1\sigma_1/y_2\} \mid \Pi_{i\in I\setminus\{1,2\}} R_i\sigma_i) \\ \text{(Lemma 2)} &\equiv \nu\tilde{a}.(R'_1\sigma_1 \mid \nu\tilde{a}_2.(\Pi_{j\in J} R_{2,j}) \mid \Pi_{i\in I\setminus\{1,2\}} R_i\sigma_i) \\ \text{(Scope extr., disj. } \tilde{a}_2) &\equiv \nu\tilde{a}, \tilde{a}_2.(R'_1\sigma_1 \mid \Pi_{j\in J} R_{2,j} \mid \Pi_{i\in I\setminus\{1,2\}} R_i\sigma_i) \\ \text{(Argumentation)} &= \nu\tilde{a}, \tilde{a}_2.(R'_1\sigma_1 \mid \Pi_{j\in J} P_{2,j}\sigma_2\{y_1\sigma_1/y_2\} \mid \Pi_{i\in I\setminus\{1,2\}} R_i\sigma_i). \end{aligned}$$

To show $\sigma_2\{y_1\sigma_1/y_2\} : fn(P_{2,j}) \to fn(P) \cup \tilde{a} \cup \tilde{a}_2$, we establish the inclusion $fn(R_{2,j}) = fn(P_{2,j}\sigma_2\{y_1\sigma_1/y_2\}) \subseteq fn(P) \cup \tilde{a} \cup \tilde{a}_2$. A name $a \in fn(R_{2,j})$, which is not in $\tilde{a}_2$, is in $fn(\nu\tilde{a}_2.(\Pi_{j\in J} R_{2,j}))$. Structural congruence yields the equality $fn(\nu\tilde{a}_2.(\Pi_{j\in J} R_{2,j})) = fn(R'_2\sigma_2\{y_1\sigma_1/y_2\})$. By the hypothesis, σ_2 maps $fn(R'_2)$ into $fn(P) \cup \tilde{a}$ and σ_1 maps y_1 into $fn(P) \cup \tilde{a}$. So the range of $\sigma_2\{y_1\sigma_1/y_2\}$ is correct. We handle $R_1\sigma_1$ similarly and get the following process that is structurally congruent with Q_{n+1} and of the desired form:

$$\nu\tilde{a}, \tilde{a}_1, \tilde{a}_2.(\Pi_{j\in J_1} P_{1,j}\sigma_1 \mid \Pi_{j\in J_2} P_{2,j}\sigma_2\{y_1\sigma_1/y_2\} \mid \Pi_{i\in I\setminus\{1,2\}} R_i\sigma_i).$$

Let $F \in Frag(\nu(Q))$. The first part of the proposition gives $Q \equiv \nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i)$ with $Q_i \in derivatives(P)$ and $\sigma_i : fn(Q_i) \to fn(P) \cup \tilde{a}$. With Proposition 1 we have $\nu(Q) \widehat{\equiv} \nu(\nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i))$, i.e., $F \equiv G$ with $G \in Frag(\nu(\nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i)))$. Computing the restricted form does not change the sequential processes:

$$\mathcal{S}(G) \subseteq \mathcal{S}(\nu(\nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i))) = \mathcal{S}(\nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i)) = \bigcup_{i\in I}\{Q_i\sigma_i\}.$$

Lemma 2 gives $G \equiv \nu\tilde{a}'.(\Pi_{j\in J} G_j)$ with $G_j \in \mathcal{S}(G) \subseteq \bigcup_{i\in I}\{Q_i\sigma_i\}$. So we have $F \equiv G \equiv \nu\tilde{a}'.(\Pi_{j\in J} Q_j\sigma_j)$ with $Q_j \in derivatives(P)$.

To see that the range of the σ_j is $fn(P) \bigcup \tilde{a}'$, consider a name $a \in fn\left(Q_j\sigma_j\right)$ with $a \notin \tilde{a}'$. Then we have $a \in fn\left(\nu\tilde{a}'.(\Pi_{j\in J} Q_j\sigma_j)\right) = fn(G)$ with the invariance of $fn(-)$ under structural congruence. Since G is a fragment in $\nu(\nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i))$ we have $fn(G) \subseteq fn(\nu(\nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i))) = fn(\nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i))$. The definition of $fn(-)$ yields $fn(\nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i)) = (\bigcup_{i\in I} fn(Q_i\sigma_i)) \setminus \tilde{a} \subseteq fn(P)$. The inclusion holds with $fn(Q_i\sigma_i) \subseteq fn(P) \cup \tilde{a}$ since $\sigma_i : fn(Q_i) \to fn(P) \cup \tilde{a}$ and so $fn(Q_i\sigma_i)) \setminus \tilde{a} \subseteq fn(P)$. This shows $a \in fn(P)$ and so the range of σ_j is $fn(P) \cup \tilde{a}'$, as required. □

Proof of Theorem 2 We first show that boundedness of $number_{|}(F)$ implies structural stationarity. Consider process $P \in \mathcal{P}$, where $k_s \in \mathbb{N}$ is a bound on the number of sequential processes in all reachable fragments.

To begin with, we argue that FG_j is finite for every $j \in \mathbb{N}$. The finiteness of FG follows immediately. There are finitely many derivatives $P' \in derivatives(P)$ by Lemma 11. Every process has finitely many free names. Thus, $fn\left(P'\right)$ and $fn(P)$ are finite. The set $u_1, \ldots, u_{j\cdot \max FN}$ also is finite. This shows there are finitely many mappings $\sigma : fn\left(P'\right) \to fn(P) \cup \tilde{u}_j$ for every P'. We conclude FG_j is finite. It remains to be shown that up to structural congruence every reachable fragment is included in FG.

Consider $Q \in Reach(P)$ and $F \in Frag(\nu(Q))$. With Proposition 3, fragment F is structurally congruent with $\nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i)$, where $Q_i \in derivatives(P)$ and $\sigma_i : fn(Q_i) \to fn(P) \cup \tilde{a}$. We show that

1. F is structurally congruent with a fragment in $FG_{|I|}$ and
2. $|I| \leq k_s$.

This proves the theorem as it shows that F is structurally congruent with a fragment in FG. Let $\tilde{a}' = \tilde{a} \cap fn(\Pi_{i\in I} Q_i\sigma_i)$ be the names that are actually used in the $Q_i\sigma_i$. We need that their number is bounded by $|I| \cdot \max FN$:

$$\begin{aligned}
& |\tilde{a}'| \\
(\text{Definition } \tilde{a}') \quad & \leq |fn(\Pi_{i\in I} Q_i\sigma_i)| \\
(\text{Definition } fn(-)) \quad & = |\textstyle\bigcup_{i\in I} fn(Q_i\sigma_i)| \\
& \leq \Sigma_{i\in I} |fn(Q_i\sigma_i)| \\
(\sigma_i \text{ may identify names}) \quad & \leq \Sigma_{i\in I} |fn(Q_i)| \\
(\text{Definition } \max FN) \quad & \leq \Sigma_{i\in I} \max FN \\
& = |I| \cdot \max FN.
\end{aligned}$$

With the congruence $\nu a.P \equiv P$ if $a \notin fn(P)$, we remove the names in $\tilde{a} \setminus \tilde{a}'$ from $\nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i)$. With α-conversion we rename the names in $\tilde{a}'$ to names in $\tilde{u}_{|I|}$. Since

$|\tilde{a}'| \leq |I| \cdot \max FN$, we add the missing names in $\tilde{u}_{|I|}$, again using $\nu a.P \equiv P$ if $a \notin fn(P)$:

$$\begin{aligned} & \nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i) \\ (\text{Def. } \tilde{a}') &\equiv \nu\tilde{a}'.(\Pi_{i\in I} Q_i\sigma_i) \\ (\tilde{a}' = a_1, \ldots, a_l) &\equiv \nu u_1, \ldots, u_l.(\Pi_{i\in I} Q_i\sigma_i\{u_l/a_l\}\ldots\{u_1/a_1\}). \end{aligned}$$

Define the substitutions σ_i' by $a\sigma_i' := a\sigma_i\{u_l/a_l\}\ldots\{u_1/a_1\}$:

$$\begin{aligned} & = \nu u_1, \ldots, u_l.(\Pi_{i\in I} Q_i\sigma_i') \\ (l \leq |I| \cdot \max FN, \text{def. } \tilde{u}_{|I|}) &\equiv \nu\tilde{u}_{|I|}.(\Pi_{i\in I} Q_i\sigma_i'). \end{aligned}$$

We now have $F \equiv \nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i) \equiv \nu\tilde{u}_{|I|}.(\Pi_{i\in I} Q_i\sigma_i')$. With Lemma 4 and Proposition 1 it follows:

$$F = \nu(F) \mathrel{\widehat{\equiv}} \nu\left(\nu\tilde{u}_{|I|}.(\Pi_{i\in I} Q_i\sigma_i')\right).$$

As F is a fragment, restricted equivalence $\mathrel{\widehat{\equiv}}$ implies $\nu\left(\nu\tilde{u}_{|I|}.(\Pi_{i\in I} Q_i\sigma_i')\right)$ is a fragment and thus in $FG_{|I|}$. With $|I| \leq k_s$ the inclusion $FG_{|I|} \subseteq FG$ holds:

$$\begin{aligned} & k_s \\ (\text{Assumption}) &\geq number_{|}(F) \\ (number_{|}(-) \text{ invariant under } \equiv) &= number_{|}(\nu\tilde{a}.(\Pi_{i\in I} Q_i\sigma_i)) \\ (\text{Definition } number_{|}(-)) &= \Sigma_{i\in I} number_{|}(Q_i\sigma_i) \\ (number_{|}(-) \text{ invariant under } \sigma) &= \Sigma_{i\in I} number_{|}(Q_i) \\ (\text{Definition } derivatives(-)) &= |I|. \end{aligned}$$

This concludes the proof of the implication from right to left. Vice versa, if P is structurally stationary, all reachable processes are made up of finitely many fragments $F_1, \ldots, F_n$. Thus, the number of sequential processes in all reachable fragments is bounded by

$$\max\{number_{|}(F_i) \mid 1 \leq i \leq n\}.$$

Proof of Lemma 12 By $number_{|}(K\lfloor\tilde{a}\rfloor) = 1 = 1^0 = nest_{|}(K\lfloor\tilde{a}\rfloor)^{nest_\nu(K\lfloor\tilde{a}\rfloor)}$, the inequality holds in the base case $K\lfloor\tilde{a}\rfloor$. The proof for $\Sigma_{i\in I\neq\emptyset}\pi_i.P_i$ is analogue.

To show the induction step, we assume that $number_{|}(F_i) \leq nest_{|}(F_i)^{nest_\nu(F_i)}$ holds for all F_i with $1 \leq i \leq n$. We then have for $F = \nu a.(F_1 \mid \ldots \mid F_n)$:

$$\begin{aligned} & number_{|}(F) \\ (\text{Def. } number_{|}) &= \Sigma_{i=1}^n number_{|}(F_i) \\ (\text{Hypothesis}) &\leq \Sigma_{i=1}^n nest_{|}(F_i)^{nest_\nu(F_i)} \\ (\text{Def. max}) &\leq \Sigma_{i=1}^n \max\{nest_{|}(F_i) \mid 1 \leq i \leq n\}^{\max\{nest_\nu(F_i) \mid 1\leq i\leq n\}}. \end{aligned}$$

With $\max_| := \max\{nest_| (F_i) \mid 1 \leq i \leq n\}$ we continue:

$$\begin{aligned}
&= \Sigma_{i=1}^{n} \max_|^{\max\{nest_\nu(F_i) \mid 1 \leq i \leq n\}} \\
&= n \cdot \max_|^{\max\{nest_\nu(F_i) \mid 1 \leq i \leq n\}} \\
\text{(Def. max)} \quad &\leq \max\{n, \max_|\} \cdot \max\{n, \max_|\}^{\max\{nest_\nu(F_i) \mid 1 \leq i \leq n\}} \\
&= \max\{n, \max_|\}^{1+\max\{nest_\nu(F_i) \mid 1 \leq i \leq n\}} \\
\text{(Def. } nest_\nu (F)) \quad &= \max\{n, \max_|\}^{nest_\nu(F)} \\
\text{(Def. } \max_|) \quad &= \max\{n, nest_| (F_1), \ldots, nest_| (F_n)\}^{nest_\nu(F)} \\
\text{(Def. } nest_| (F)) \quad &= nest_| (F)^{nest_\nu(F)}.
\end{aligned}$$

□

Proof of Theorem 3 If the process is structurally stationary, there is a finite set of fragments $\{F_1, \ldots, F_n\}$ so that every reachable fragment is structurally congruent with some F_i. Then $\max\{depth(F_i) \mid 1 \leq i \leq n\}$ is a bound on the depth and $\max\{breadth(F_i) \mid 1 \leq i \leq n\}$ is a bound on the breadth of all reachable fragments.

Vice versa, if we assume boundedness in breadth and depth there are k_b and k_d so that for all $Q \in Reach(P)$ and all $F \in Frag(\nu(Q))$ we have $breadth(F) \leq k_b$ and $depth(F) \leq k_d$. We prove boundedness in the number of sequential processes:

$$\exists k_s \in \mathbb{N} : \forall Q \in Reach(P) : \forall F \in Frag(\nu(Q)) : number_| (F) \leq k_s.$$

We claim that $k_b^{k_d}$ is a bound on the number of sequential processes. Consider $Q \in Reach(P)$ and $F \in Frag(\nu(Q))$. For every fragment F, there is $F_D \equiv F$ so that $nest_\nu(F_D) = \min\{nest_\nu(F') \mid F' \equiv F\} = depth(F)$. For this F_D the inequality $nest_| (F_D) \leq \max\{nest_|(F') \mid F' \equiv F\} = breadth(F)$ holds. We compute

$$\begin{aligned}
& number_| (F) \\
(number_| (-) \text{ invariant under } \equiv) \quad &= number_| (F_D) \\
\text{(Lemma 12)} \quad &\leq nest_| (F_D)^{nest_\nu(F_D)} \\
\text{(Observation on } nest_| (F_D)) \quad &\leq breadth(F)^{nest_\nu(F_D)} \\
\text{(Observation on } nest_\nu (F_D)) \quad &= breadth(F)^{depth(F)} \\
(k_b \text{ and } k_d \text{ bounds on breadth and depth}) \quad &\leq k_b^{k_d}.
\end{aligned}$$

This proves P is bounded in the number of sequential processes. With Theorem 2 it follows that P is structurally stationary. □

Appendix D: Proofs of Section 6

Proof of Lemma 14 We establish the statement for $PT \in \mathcal{P}_{\mathcal{PT}}$, the proofs for $HD \in \mathcal{P}_{\mathcal{HD}}$ and $FH \in \mathcal{P}_{\mathcal{FH}}$ are similar.

In the base case, we consider the processes $K_P\lfloor\tilde{p}\rfloor$ and $\nu\tilde{a}.(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i)$ with $fn\left(\nu\tilde{a}.(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i)\right) \subseteq \mathcal{N}_\mathcal{P}$. The restricted form does not change a process identifier, i.e., $\nu(K_P\lfloor\tilde{p}\rfloor) = K_P\lfloor\tilde{p}\rfloor$. Thus $Frag(\nu(K_P\lfloor\tilde{p}\rfloor)) = \{K_P\lfloor\tilde{p}\rfloor\}$ which is a fragment in *fhf*. In case of a choice composition, the restricted form removes those names from $\tilde{a}$ that are not in $fn\left(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i\right)$. Thus we have $Frag\left(\nu\left(\nu\tilde{a}.(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i)\right)\right) =$

$\{\nu\tilde{a}'.(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i)\}$ with $\tilde{a}' \subseteq \tilde{a}$. As only superfluous restrictions are removed, the free names remain in $\mathcal{N}_\mathcal{P}$ and we have a fragment in *fhf*.

We assume that all fragments $F \in Frag(\nu(PT_i))$ are in *fhf*. By definition, $\nu(PT_1 \mid PT_2) = \nu(PT_1) \mid \nu(PT_2)$. Thus, $F \in Frag(\nu(PT_1 \mid PT_2))$ if and only if $F \in Frag(\nu(PT_1))$ or $F \in Frag(\nu(PT_2))$. In both cases, the hypothesis yields that F is in *fhf*. □

Proof of Lemma 15 We prove the first two statements, the proof of the third relies on the insights from the preceding proofs.

Consider fragment F in *fhf* with $number_|(F) > 1$. According to Definition 23, F is structurally congruent with $\nu\tilde{a}.(SQ_1 \mid \ldots \mid SQ_n \mid CN)$, where we wlog. assume a process CN. There are three possibilities for F to perform a reaction. A process $SQ = \tau.SQ' + M$ or the process CN consumes a τ prefix, a process $K_S\lfloor\tilde{a}\rfloor$ calls its definition, or two processes SQ_1 and SQ_2 or a process SQ and CN communicate.

Consider the first case where $SQ_1 = \tau.SQ_1' + M$ consumes a τ prefix, i.e., we get $\nu\tilde{a}.(\tau.SQ_1' + M \mid \Pi_{i=2}^n SQ_i \mid CN) \to \nu\tilde{a}.(SQ_1' \mid \Pi_{i=2}^n SQ_i \mid CN)$. Since the free names in M are lost we have $fn(SQ_1') \subseteq fn(SQ_1)$. Consequently, the restricted form of the resulting process may consist of several fragments, $\nu\big(\nu\tilde{a}.(SQ_1' \mid \Pi_{i=2}^n SQ_i \mid CN)\big) = \Pi_{j\in J} F_j$.

We first consider F_j with $number_|(F_j) > 1$ and show that it is in *fhf*. Since there is at most one process CN in F, there is at most one process CN in F_j. The remaining processes in F as well as SQ_1' are processes SQ. Thus, F_j is a parallel composition of processes SQ with at most one process CN. Since $fn(F_j) \subseteq fn(\nu\tilde{a}.(SQ_1' \mid \Pi_{i=2}^n SQ_i \mid CN)) \subseteq fn(F) \subseteq \mathcal{N}_\mathcal{P}$, the free names of F_j are in $\mathcal{N}_\mathcal{P}$. We prove the inequality to conclude F_j is in *fhf*:

$$
\begin{aligned}
& number_|\left(F_j\right) + \#_R(F_j) \\
(\#_R(F_j) = \#_R(F) \text{ if } CN \text{ in } F_j, 0 \text{ otherwise}) & \leq number_|\left(F_j\right) + \#_R(F) \\
(number_|\left(F_j\right) \leq number_|(F)) & \leq number_|(F) + \#_R(F) \\
(\text{Assumption}) & \leq \max_R(FH) + 1.
\end{aligned}
$$

Consider a fragment (F_j) with $number_|(F_j) = 1$. Then F_j is either a process SQ or the process CN. With the inclusion $fn(F_j) \subseteq fn(F) \subseteq \mathcal{N}_\mathcal{P}$ and the fact that SQ does not contain names in $\mathcal{N}_\mathcal{P}$ we get $fn(SQ) = \emptyset$, which means SQ is in *fhf*. This also shows $fn(CN) \subseteq \mathcal{N}_\mathcal{P}$. By the assumption that F is in *fhf* we have $number_|(F) + \#_R(F) \leq \max_R(FH) + 1$. Since only CN contains receiving prefixes, the equality $\#_R(F) = \#_R(CN)$ holds. We conclude $number_|(F) + \#_R(CN) \leq \max_R(FH) + 1$, which is equivalent with $\#_R(CN) \leq \max_R(FH) + 1 - number_|(F)$. Since $number_|(F) > 1$, the inequality $\#_R(CN) < \max_R(FH)$ holds, i.e., CN is in *fhf*.

Consider $\nu\tilde{c}.(p(x).CN + M) \mid \nu\tilde{a}.(\overline{p}\langle a\rangle.SQ + N) \to \nu\tilde{a}.(\nu\tilde{c}.(CN\{a/x\}) \mid SQ)$. If $a \notin fn(CN\{a/x\})$ ($a \notin fn(SQ)$ is similar), the restricted form consists of two fragments, $\nu(\nu\tilde{a}.(\nu\tilde{c}.(CN\{a/x\}) \mid SQ)) = \nu(\nu\tilde{c}.CN) \mid \nu(\nu\tilde{a}.SQ)$. Like in the first part of the lemma, we get $fn(\nu\tilde{c}.CN) \subseteq \mathcal{N}_\mathcal{P}$ and $fn(\nu\tilde{a}.SQ) \subseteq \mathcal{N}_\mathcal{P}$. Again an SQ process does not use names in $\mathcal{N}_\mathcal{P}$, which means $fn(\nu\tilde{a}.SQ) = \emptyset$. The fragment is in *fhf*. To show $\nu\tilde{c}.CN$ is also in *fhf*, we establish the required inequality as follows:

$$\#_R(\nu\tilde{c}.CN) < \#_R(\nu\tilde{c}.(p(x).CN + M)) \leq \max_R(FH).$$

If $a \in fn(CN\{a/x\}) \cap fn(SQ)$, the fragment has one SQ and one CN process. We show that the free names are in $\mathcal{N}_\mathcal{P}$. From $a \in \tilde{a}$ we derive the inclusion $fn(\nu\tilde{c}.(CN\{a/x\})) \setminus \tilde{a} \subseteq fn(\nu\tilde{c}.(p(x).CN + M)) \subseteq \mathcal{N}_\mathcal{P}$. Similarly, we show that $fn(SQ) \setminus \tilde{a} = \emptyset$. Hence, the free

names in $\nu\tilde{a}.(\nu\tilde{c}.(CN\{a/x\}) \mid SQ)$ are in $\mathcal{N}_\mathcal{P}$. We prove the inequality as follows:

$$\begin{aligned} & number_{\mid}(\nu\tilde{a}.(\nu\tilde{c}.(CN\{a/x\}) \mid SQ)) + \\ & \#_R(\nu\tilde{a}.(\nu\tilde{c}.(CN\{a/x\}) \mid SQ)) \\ (\text{Def. } number_{\mid}(-)) &= 2 + \#_R(\nu\tilde{a}.(\nu\tilde{c}.(CN\{a/x\}) \mid SQ)) \\ (\text{Def. } SQ) &= 2 + \#_R(\nu\tilde{c}.(CN\{a/x\})) \\ (\text{Def. } \#_R(-)) &\leq 2 + \#_R(\nu\tilde{c}.(p(x).CN + M)) - 1 \\ (\text{Assumption}) &\leq 1 + \max_R(FH). \end{aligned}$$

This concludes the proof. □

Proof of Lemma 13 Let $FH \in \mathcal{P}_{\mathcal{FH}}$. The base case is $Q_0 = FH$, i.e., the reaction sequence of length zero. By Lemma 14 all fragments in $Frag(\nu(Q_0))$ are in *fhf*.

For the induction step, we assume all fragments in $Frag(\nu(Q_n))$ are in *fhf*, where Q_n is reachable from FH with $n \in \mathbb{N}$ reactions. We have

$$\begin{aligned} Q_n \to Q_{n+1} \Leftrightarrow\ & \exists F \in Frag(\nu(Q_n)), R, Q'_n \in \mathcal{P} : F \to R \\ & \wedge Q_n \equiv F \mid Q'_n \wedge Q_{n+1} \equiv R \mid Q'_n \\ \vee\ & \exists F_1, F_2 \in Frag(\nu(Q_n)), R, Q'_n \in \mathcal{P} : F_1 \mid F_2 \to R \\ & \wedge Q_n \equiv F_1 \mid F_2 \mid Q'_n \wedge Q_{n+1} \equiv R \mid Q'_n. \end{aligned}$$

We distinguish the cases and show that the fragments $F \in Frag(\nu(R))$ are in *fhf*. This proves the induction step as $F \in Frag\left(\nu\left(Q'_n\right)\right)$ is in *fhf* by the hypothesis.

We begin with reactions $F \to R$. For calling process identifiers, Lemma 14 shows that the fragments in R are in *fhf*.

By Definition 20, $CN = \nu\tilde{a}.(\Sigma_{i\in I}.\pi_i^R.CN_i)$. The process consumes τ prefixes, i.e., $CN \to \nu\tilde{a}.CN_i$ if and only if $\pi_i^R = \tau$. With $fn(\nu\tilde{a}.CN_i) \subseteq fn(CN) \subseteq \mathcal{N}_\mathcal{P}$ the free names are in $\mathcal{N}_\mathcal{P}$. With the definition of $\#_R(-)$ and the hypothesis we derive $\#_R(\nu\tilde{a}.CN_i) \leq \#_R(CN) \leq \max_R(FH)$. Thus, $\nu\tilde{a}.CN_i$ is in *fhf*.

Similarly, $\Sigma_{i\in I}\pi_i.FH_i \to FH_i$ if and only if $\pi_i = \tau$. It follows from Lemma 14 that the fragments in $Frag(\nu(FH_i))$ are in *fhf*.

Consider SQ with $fn(SQ) = \emptyset$. According to Definition 19, there are two cases. If $SQ = \nu\tilde{a}.(\Sigma_{i\in I}\pi_i.SQ_i)$ we have $SQ \to \nu\tilde{a}.SQ_i$ if and only if $\pi_i = \tau$. Furthermore, $fn\left(\nu\tilde{a}.SQ_i\right) \subseteq fn(SQ) = \emptyset$. This proves the *fhf*. In the second case, we have a process identifier $SQ = \nu\tilde{a}.K_S\lfloor\tilde{a}'\rfloor$ with $K_S(\tilde{x}) := SQ'$. This yields the reaction $SQ \to \nu\tilde{a}.(SQ'\{\tilde{a}'/\tilde{x}\})$. Since we require $fn\left(SQ'\right) \subseteq \tilde{x}$, the inclusion $fn\left(\nu\tilde{a}.(SQ'\{\tilde{a}'/\tilde{x}\})\right) \subseteq \tilde{a}' \setminus \tilde{a} = fn(SQ) = \emptyset$ holds. The fragment is in *fhf*.

Consider fragment F with $number_{\mid}(F) > 1$ and $F \to R$. By the hypothesis F is in *fhf*. Lemma 15 shows that R is a parallel composition of fragments in *fhf*.

Fragment $\nu\tilde{a}.(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i)$ has no reactions, which proves the case.

Let $F_1 \mid F_2 \to R$. An identifier or an SQ process where all names are restricted does not have a reaction with a second fragment. Let $F_1 = \nu\tilde{a}.(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i)$. This fragments sends on a distinguished channel. The only fragments that listen on these channels are fragments CN and fragments F_2 with $number_{\mid}(F_2) > 1$. Consider $CN \mid \nu\tilde{a}.(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i) \to R$. By the hypothesis, CN is in *fhf*. Lemma 15 shows that R is a parallel composition of fragments in *fhf*. The case $F \mid \nu\tilde{a}.(\Sigma_{i\in I}\overline{p_i}\langle a_i\rangle.SQ_i)$ is similar.

A fragment $\Sigma_{i\in I}\pi_i.FH_i = \overline{a}\langle b\rangle.FH + M$ sends on a public channel. Only fragments of the same form listen on public channels. Thus, we have a reaction $\overline{a}\langle b\rangle.FH + M \mid a(x).FH' + N$

$\rightarrow FH \mid FH'\{b/x\}$. For the resulting fragments, it holds $F \in Frag\left(\nu\left(FH \mid FH'\{b/x\}\right)\right)$ if and only if $F \in Frag\left(\nu(FH)\right)$ or $F \in Frag\left(\nu\left(FH'\{b/x\}\right)\right)$. In both cases, Lemma 14 states that F is in *fhf*. □

References

1. Amadio, R. M., Meyssonnier, C.: On decidability of the control reachability problem in the asynchronous π-calculus. Nord. J. Comput. **9**(1), 70–101 (2002)
2. Bodei, C., Degano, P., Nielson, F., Riis Nielson, H.: Control flow analysis for the π-calculus. In: Proc. of the 9th International Conference on Concurrency Theory, CONCUR. LNCS, vol. 1466, pp. 84–98. Springer, Heidelberg (1998)
3. Busi, N., Gorrieri, R.: A Petri net semantics for π-calculus. In: Proc. of the 6th International Conference on Concurrency Theory, *CONCUR*. LNCS, vol. 962, pp. 145–159. Springer, Heidelberg (1995)
4. Busi, N., Gorrieri, R.: Distributed semantics for the π-calculus based on Petri nets with inhibitor arcs. Journal of Logic and Algebraic Programming, 46 pp. (2008, to appear)
5. Busi, N.: Analysis issues in Petri nets with inhibitor arcs. Theor. Comput. Sci. **275**(1–2), 127–177 (2002)
6. Caires, L.: Behavioural and spatial observations in a logic for the π-Calculus. In: Proc. of the 7th International Conference on Foundations of Software Science and Computation Structures, FOSSACS. LNCS, vol. 2987, pp. 72–89. Springer, Heidelberg (2004) SPATIAL LOGIC MODEL CHECKER: http://ctp.di.fct.unl.pt/SLMC/
7. Caires, L., Cardelli, L.: A spatial logic for concurrency (part I). Inf. Comput. **186**(2), 194–235 (2003)
8. Clarke, E. M., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. In: Proc. of the 12th International Conference on Computer Aided Verification, CAV. LNCS, vol. 1855, pp. 154–169. Springer, Heidelberg (2000)
9. Clarke, E. M., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (1999)
10. Dam, M.: Model checking mobile processes. Inf. Comput. **129**(1), 35–51 (1996)
11. Devillers, R., Klaudel, H., Koutny, M.: A Petri net semantics of the finite π-Calculus terms. Fundam. Inf. **70**(3), 203–226 (2006)
12. Devillers, R., Klaudel, H., Koutny, M.: A Petri net translation of π-Calculus terms. In: Proc. of the 3rd International Colloquium on Theoretical Aspects of Computing, ICTAC. LNCS, vol. 4281, pp. 138–152. Springer, Heidelberg (2006)
13. Engelfriet, J., Gelsema, T.: Multisets and structural congruence of the pi-calculus with replication. Theor. Comput. Sci. **211**(1-2), 311–337 (1999)
14. Engelfriet, J., Gelsema, T.: The decidability of structural congruence for replication restricted pi-calculus processes. Technical report, Leiden Institute of Advanced Computer Science (2004). Revised 2005
15. Engelfriet, J., Gelsema, T.: A new natural structural congruence in the pi-calculus with replication. Acta Inf. **40**(6), 385–430 (2004)
16. Engelfriet, J., Gelsema, T.: An exercise in structural congruence. Inf. Process. Lett. **101**(1), 1–5 (2007)
17. Engelfriet, J.: A multiset semantics for the pi-calculus with replication. Theor. Comput. Sci. **153**(1–2), 65–94 (1996)
18. Esparza, J., Römer, S., Vogler, W.: An improvement of McMillan's unfolding algorithm. Formal Methods Syst. Des. **20**(3), 285–310 (2002)
19. Esparza, J., Schröter, C.: Net reductions for LTL model-checking. In: Proc. of the 11th Advanced Research Working Conference on Correct Hardware Design and Verification Methods, CHARME. LNCS, vol. 2144, pp. 310–324. Springer, Heidelberg (2001)
20. Esparza, J.: Decidability of model checking for infinite-state concurrent systems. Acta Inf. **34**(2), 85–107 (1997)
21. Esparza, J.: Petri Nets, commutative context-free grammars, and basic parallel processes. Fundam. Inf. **31**(1), 13–25 (1997)
22. Ferrari, G.-L., Gnesi, S., Montanari, U., Pistore, M.: A model-checking verification environment for mobile processes. ACM Trans. Softw. Eng. Methodol. **12**(4):440–473 (2003). HAL: http://fmt.isti.cnr.it:8080/hal/
23. Hsu, A., Eskafi, F., Sachs, S., Varaiya, P.: Design of platoon maneuver protocols for ivhs. Path research report, Institute of Transportation Studies, University of California, Berkeley (1991)
24. Jančar, P.: Undecidability of bisimilarity for Petri nets and some related problems. Theor. Comput. Sci. **148**(2), 281–301 (1995)
25. Khomenko, V.: Model Checking Based on Prefixes of Petri Net Unfoldings. PhD thesis, School of Computing Science, Newcastle University (2003)

26. Khomenko, V., Koutny, M., Niaouris, A.: Applying Petri net unfoldings for verification of mobile systems. In: Proc. of the 4th Workshop on Modelling of Objects, Components and Agents, MOCA. Bericht FBI-HH-B-267/06, pp. 161–178. University of Hamburg (2006)
27. Khomenko, V., Meyer, R.: Checking π-Calculus structural congruence is graph isomorphism complete. Technical Report CS-TR-1100, School of Computing Science, Newcastle University (2008). URL: http://www.cs.ncl.ac.uk/publications/trs/abstract/1100
28. McMillan, K.: Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In: Proc. of the 4th International Workshop on Computer Aided Verification, CAV. LNCS, vol. 663, pp. 164–174. Springer, Heidelberg (1992)
29. Meyer, R.: On boundedness in depth in the π-calculus. In: Proc. of the 5th IFIP International Conference on Theoretical Computer Science, IFIP TCS. IFIP, vol. 273, pp. 477–489. Springer, Heidelberg (2008)
30. Milner, R.: Communicating and Mobile Systems: the π-Calculus. Cambridge University Press, London (1999)
31. Meyer, R., Khomenko, V., Strazny, T.: A practical approach to verification of mobile systems using net unfoldings. In: Proc. of the 29th International Conference on Application and Theory of Petri Nets and Other Models of Concurrency, ATPN. LNCS, vol. 5062, pp. 327–347. Springer, Heidelberg (2008)
32. Montanari, U., Pistore, M.: Checking bisimilarity for finitary π-calculus. In: Proc. of the 6th International Conference on Concurrency Theory, CONCUR. LNCS, vol. 962, pp. 42–56. Springer, Heidelberg (1995)
33. Montanari, U., Pistore, M.: Concurrent semantics for the π-calculus. Electr. Notes Theor. Comput. Sci. **1**, 411–429 (1995)
34. Montanari, U., Pistore, M.: History dependent automata. Technical report, Instituto Trentino di Cultura (2001)
35. Olderog, E.-R.: Nets, Terms and Formulas: Three Views of Concurrent Processes and Their Relationship. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, London (1991)
36. Pistore, M.: History Dependent Automata. PhD thesis, Dipartimento di Informatica, Università di Pisa (1999)
37. Reisig, W.: Petri nets: an introduction. Monographs in Theoretical Computer Science. An EATCS Series. Springer, Heidelberg (1985)
38. Strazny, T., Meyer, R.: PETRUCHIO homepage. http://petruchio.informatik.uni-oldenburg.de (2008)
39. Strazny, T.: Entwurf und Implementierung von Algorithmen zur Berechnung von Petrinetz-Semantiken für Pi-Kalkül-Prozesse. Master's thesis, Department of Computing Science, University of Oldenburg (2007)
40. Sangiorgi, D., Walker, D.: The π-calculus: a Theory of Mobile Processes. Cambridge University Press, London (2001)
41. Victor, B., Moller, F.: The mobility workbench: a tool for the π-calculus. In: Proc. of the 6th International Conference on Computer Aided Verification. LNCS, vol. 818, pp. 428–440. Springer, Heidelberg (1994), MOBILITY WORKBENCH: http://www.it.uu.se/research/group/mobility/mwb