Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A network-conscious π-calculus and its coalgebraic semantics ☆

Ugo Montanari, Matteo Sammartino *

Dipartimento di Informatica, Università di Pisa, Largo Bruno Pontecorvo 3, 56127 Pisa, Italy

ARTICLE INFO

Available online xxxx

Keywords:
π-calculus
Network-awareness
Presheaves
Coalgebras
Concurrent semantics

ABSTRACT

Traditional process calculi usually abstract away from network details, modeling only communication over shared channels. They, however, seem inadequate to describe new network architectures, such as Software Defined Networks, where programs are allowed to manipulate the infrastructure. In this paper we present the *Network Conscious π-calculus* (NCPi), a proper extension of the π-calculus with an explicit notion of network: network links and nodes are represented as names, in full analogy with ordinary π-calculus names, and observations are routing paths through which data is transported. However, restricted links do not appear in the observations, which thus can possibly be as abstract as in the π-calculus. Then we construct a presheaf-based coalgebraic semantics for NCPi along the lines of Turi–Plotkin's approach, by indexing processes with the network resources they use: we give a model for observational equivalence in this context, and we prove that it admits an equivalent nominal automaton (HD-automaton), suitable for verification. Finally, we give a concurrent semantics for NCPi where observations are multisets of routing paths. We show that bisimilarity for this semantics is a congruence, and this property holds also for the concurrent version of the π-calculus.

1. Introduction

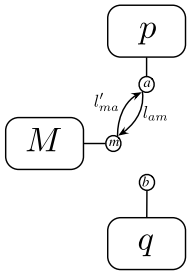
The trend in networking is going towards more "open" architectures, where the infrastructure can be manipulated in software. This trend started in the nineties, when OpenSig [1] and Active Networks [2] were presented, but neither gained wide acceptance due to security and performance problems. More recently, OpenFlow [3,4] or, more broadly, Software Defined Networking has become the leading approach, supported by Google, Facebook, Microsoft and others. Software defined networks (SDNs) are networks in which a programmable controller machine manages a group of switches, by instructing them to install or uninstall forwarding rules and report traffic statistics.

Traditional process calculi, such as π-calculus [5], CCS [6] and others, seem inadequate to describe these kinds of networks, because they abstract away from network details. In fact, two processes are allowed to communicate only through shared channels and it is not possible to express explicitly the fact that there is some complex connector between them. To give better visibility to the network architecture, in recent years network-aware extensions of known calculi have been devised [7,8].

☆ Research supported by the EU Integrated Project 257414 ASCENS and by the Italian MIUR Project CINA (PRIN 2010), grant number 2010LHT4KM.
* Corresponding author. Tel.: +39 0502213117.
E-mail addresses: ugo@di.unipi.it (U. Montanari), sammarti@di.unipi.it (M. Sammartino).
URLs: http://www.di.unipi.it/~ugo (U. Montanari), http://www.di.unipi.it/~sammarti (M. Sammartino).

http://dx.doi.org/10.1016/j.tcs.2014.03.009
0304-3975/

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009



Processes

$$M \stackrel{\text{def}}{=} m(x).m(y).(l_{xy})(\overline{m}l_{xy}.M)$$
$$p \stackrel{\text{def}}{=} \overline{a}a.\overline{a}b.a(l_{ab}).(\overline{a}c.p' \,|\, L(l_{ab}))$$
$$L(l_{xy}) \stackrel{\text{def}}{=} l_{xy}.L(l_{xy})$$
$$q \stackrel{\text{def}}{=} b(x).q'$$

System

$$S \stackrel{\text{def}}{=} p \,|\, M \,|\, q \,|\, L(l_{am}) \,|\, L(l'_{ma})$$

Fig. 1. Example system.

A network-aware π-calculus The first contribution of this paper is the *Network Conscious π-calculus* (NCPi), a seamless extension of the π-calculus with a natural notion of network: nodes and links are regarded as computational resources that can be created, passed and used to transmit, so they are represented as names, following the π-calculus methodology. The main features of NCPi are the following:

- There are two types of names: *sites*, which are the nodes of the network, and *links*, named connectors between pairs of sites. Sites are just atoms, e.g. a, links have the form l_{ab}, denoting a link named l from site a to site b.
- The syntax can express the *creation* of a link through the restriction operator, and the *activation* of a transportation service over a link through a dedicated prefix. Separating these operations agrees with the π-calculus, where creating and using a channel as subject are two distinct operations. This is different from [7,8], where pieces of network, once created, are always available.
- Observations of the labeled semantics represent transmissions in the form of routing paths.

We choose to have named connectors, instead of anonymous ones as in [7] and [8], for two main reasons. First of all, they are intended to model transportation services with distinct features (cost, bandwidth ...), which could be encoded in the label type, as we already do for the connectors' source and target. Second, this enables reusing most of the notions of the π-calculus (renaming, α-conversion, extrusion ...), suitably extended. In any case, NCPi allows one to recover anonymous connectors through the restriction operator.

One of the non-trivial aspects of NCPi is the presence of parametric names: links, in fact, are parametrized by other names, i.e. their endpoints. While this is a natural choice, it requires some care. Consider in fact a process p with a free occurrence of a link l_{ab}. If we render a unobservable, as in $(a)p$, such link would appear "dangling" to an external observer. A first solution is establishing that the link is private as well. A more elegant solution is identifying a class of well-behaved processes, where only private links can have private endpoints, i.e. $(a)(l_{ab})p$ is correct. Notice, however, that $(l_{ab})p$ is also correct: it means that the link is private, but it still can be used to connect public sites. This is analogous to what happens for free processes in a well-formed state of a CHARM [9]: their variables must belong to the global part. Another interesting aspect, unusual for π-calculus extensions, is that bisimilarity is not closed under both input prefix and parallel composition. We discuss this issue in the note added in proof.

In order to have a closer look at the calculus, consider the system in Fig. 1. Its aim is modeling a network where the routing structure is determined at run-time, like in SDNs. We have a network manager M, capable of creating new links and granting access to them, and two processes p and q, which access the network through a and b respectively; they are willing to communicate, but no links exist between a and b, so p will ask M to create such link.

The formal definition says that M can receive two sites at m, create a new link between them and send it from m. The process p can send a and b from a, wait for a link at a and then become the parallel composition of two components: the first one can send c from a; the second one can activate a transportation service over the received link, to be used by the other component. This activation is expressed as a prefix in the definition of L, which is recursive in order to simulate a persistent connection: the link prefix, in fact, expresses a single activation of the service, as input/output prefixes in the π-calculus express a single usage of their subject channel. The process q simply waits for a datum at b. Finally, the whole system S is the parallel composition of p, q, M and two processes modeling a bidirectional persistent connection between a and m.

The whole system can perform the following consecutive steps:

1. p **communicates to** M **the endpoints** a **and** b **of the link to be created:** this is the result of the (pairwise) synchronization of p, $L(l_{am})$ and M. Observations are paths, represented as lists of links plus additional information. For the involved processes we have

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

$$p \xrightarrow{\bullet;\bar{a}a} \dots \qquad L(l_{am}) \xrightarrow{a;l_{am};m} \dots \qquad M \xrightarrow{am;\bullet} \dots$$
$$p \mid L(l_{am}) \mid M \xrightarrow{\bullet;l_{am};\bullet} \dots$$

Here $\bullet;\bar{a}a$ and $am;\bullet$ express, respectively, the beginning and the end of a transmission as paths of length zero: $\bar{a}$ is the emission site, m is the reception site, and a is the datum being sent/received. $a;l_{am};m$ represents a transportation service from a to m, consisting of the only l_{am}. The concatenation of these paths yields $\bullet;l_{am};\bullet$, a complete transmission over l_{am}. As in the π-calculus, the transmitted datum, namely a, is not observable.
The state of the system after this interaction is

$$a(l_{xy}).(L(l_{xy}) \mid \bar{a}c.p') \mid (l_{ab})(\overline{m}l_{ab}.M) \mid q \mid L(l_{am}) \mid L(l'_{ma}).$$

2. M **communicates** l_{ab} **to** p**:** this happens after l_{ab} is scope extended via a suitable axiom. The resulting observation is $\bullet;l'_{ma};\bullet$ and the continuation is

$$(l_{ab})(L(l_{ab}) \mid \bar{a}c.p' \mid M \mid q) \mid L(l_{am}) \mid L(l'_{ma}).$$

3. p **communicates** c **to** q**:** in this case, despite l_{ab} is used for the transmission, only $\bullet;\bullet$ can be observed, because such link is restricted. This is analogous to the π-calculus τ-action. The continuation is

$$(l_{ab})(L(l_{ab}) \mid p' \mid M \mid q'[c/x]) \mid L(l_{am}) \mid L(l'_{ma}).$$

NCPi categorical operational semantics Processes of nominal calculi can be typed by their free names, which represent available resources for communicating with the environment. Free names can be generated, shared, transmitted and forgotten. Name constructs are very expressive, but they come with additional notions, such as α-conversion and capture avoiding substitutions, and require ad-hoc SOS rules and bisimulations that enforce name freshness. Moreover, the transition system can be infinite branching and have an infinite number of states, because names can be instantiated and allocated along transitions.

A convenient way of modeling nominal calculi, which partly solves these issues, has been presented in [10] for the paradigmatic case of the π-calculus. The basic idea is having a model where we distinguish a domain of resources, a domain of programs and a domain of "maps" between resources and programs. This kind of structure gives modularity to the model: it could allow different languages to use the same kind of resource, or the same language to use different kinds of resources. In formal terms, the domain of resources is characterized as a category $\mathbf{C}$, the domain of programs as **Set**, regarded as sets of programs, and their relationships as functors $\mathbf{C} \to$ **Set** (*presheaves on* $\mathbf{C}$). The operational semantics, then, is modeled as a coalgebra [11] with states in a presheaf, thus decorated with the amount of resources they use: this enables the explicit representation of resource allocation along transitions. Unfortunately, the state space explosion issue still exists, because resources may grow indefinitely, e.g. in recursive processes performing extrusions. However, if the presheaf of states is "well-behaved", according to [12], it is always possible to recover the minimal amount of resources a process uses, so we have a notion of deallocation. This is the key condition for the equivalence between presheaf-based coalgebras and *History Dependent (HD) automata* [13], that are automata with allocation and deallocation along transitions. HD automata admit minimal representatives, where all bisimilar states are identified, which can be computed as shown and implemented in [14].

The second contribution of this paper is the construction of two presheaf-based coalgebraic operational models for NCPi: one for observational equivalence and one for the greatest bisimulation closed under all renamings (but still not preserved by parallel composition), following and generalizing the approach of [10]. Moreover, we show that the former admits an equivalent HD-automaton. The novelty w.r.t. [10] is the treatment of complex resources, namely communication networks, where some names (links) are parametrized by other names (sites).

A concurrent semantics Interleaving semantics can be considered inadequate for distributed systems with partially asynchronous behavior, since it implicitly assumes the existence of a central arbiter who grants access to resources. This criticism is particularly relevant for NCPi.

The third contribution of this paper is a concurrent version of NCPi (κNCPi), where observations are multisets of routing paths. The main result is that bisimilarity on this concurrent semantics is a congruence. This is a desirable property for a process calculus, because it allows for the compositional analysis of systems. The authors of [8,7] treat bisimilarity and achieve compositionality as well, but they take a different approach than ours: they start from a reduction semantics, guess a suitable notion of barb, define barbed congruence by closing w.r.t. all the contexts, and then characterize it as a bisimulation equivalence on a labeled version of the transition system. In general, this approach yields labeled transition systems with succinct observations, but may resort to non-standard notions of bisimilarity, where the closure under contexts is "hardwired". We show that we can gain the congruence property through a concurrent semantics, while keeping the notion of bisimilarity as standard as possible. We exploit this result to equip the π-calculus with a concurrent and compositional semantics. The π-calculus, in fact, can be easily characterized as a syntactic restriction of κNCPi. This shows that bisimilarity not being a congruence for the ordinary π-calculus depends on the interleaving nature of the semantics, and not on the language itself. An analogous result is [15,16], but the semantics presented there allows observing the channel where a

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

synchronization is performed, whereas our concurrent semantics is closer to the π-calculus, in the sense that we adopt a synchronization mechanism that hides such a channel.

This work includes [17], where only the concurrent semantics is presented, and will be part of one of the author's Ph.D. thesis [18].

2. Preliminaries

2.1. Functor categories

Definition 2.1 *(Functor category).* Let $\mathbf{C}$ and $\mathbf{D}$ be two categories. The *functor category* $\mathbf{D}^{\mathbf{C}}$ has functors $\mathbf{C} \to \mathbf{D}$ as objects and natural transformations between them as morphisms.

Functors from any category $\mathbf{C}$ to **Set** are called (covariant) *presheaves*. A presheaf P can be intuitively seen as a family of sets indexed over the objects of $\mathbf{C}$ plus, for each $\sigma : c \to c'$, an action of σ on Pc, which we write

$$p[\sigma]_P := P\sigma(p) \quad (p \in Pc),$$

omitting the subscript P in $[\sigma]_P$ when clear from the context. This notation intentionally resembles the application of a renaming σ to a process p, namely $p\sigma$: it will, in fact, have this meaning in later chapters. The set $\int P$ of *elements* of a presheaf P is

$$\int P := \sum_{c \in |\mathbf{C}|} Pc$$

and we denote by $c \vdash p$ a pair belonging to $\int P$. Presheaf categories have the following nice property.

Property 2.2. *For any* $\mathbf{C}$, $\mathbf{Set}^{\mathbf{C}}$ *has all limits and colimits, both computed pointwise.*

An important operation on functor categories is *right Kan extension*. Consider three categories $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ and a functor $J : \mathbf{A} \to \mathbf{B}$. There is an obvious functor that turns a functor $\mathbf{B} \to \mathbf{C}$ into a functor $\mathbf{A} \to \mathbf{C}$

$$\mathscr{R} := (-) \circ J : \mathbf{C}^{\mathbf{B}} \to \mathbf{C}^{\mathbf{A}}.$$

When $\mathbf{A}$ is a subcategory of $\mathbf{B}$, then $\mathscr{R}$ simply restricts the domain of functors $\mathbf{B} \to \mathbf{C}$. Its right adjoint, which we denote by $\mathscr{E}$, performs the opposite operation: it extends the domain of a functor $\mathbf{A} \to \mathbf{C}$ to the whole $\mathbf{B}$. Conditions for its existence are given by the following statement.

Theorem 2.3. *(See [19, Corollary 2,X.3].) If* $\mathbf{A}$ *is small and* $\mathbf{C}$ *has all limits, any functor* $G : \mathbf{A} \to \mathbf{C}$ *has a right Kan extension along any* $J : \mathbf{A} \to \mathbf{B}$.

The right Kan extension of G along J can be computed, as shown in [19, X.3], as a pointwise limit. This limit, for each $b \in |\mathbf{B}|$, is constructed as follows. One considers the category whose objects are morphisms $Ja \to b$ in $\mathbf{B}$, with $a \in |\mathbf{A}|$, and whose morphisms between $f : Ja \to b$ and $g : Ja' \to b$ are morphisms $h : a \to a'$ in $\mathbf{A}$ such that Jh commutes with f and g. This is the *comma category* $b \downarrow J$. Then, one takes the diagram in $\mathbf{C}$ constructed as follows: objects are Ga, for every a such that $Ja \to b \in |b \downarrow J|$; morphisms are Gf such that f is a morphism of $b \downarrow J$. Notice that, in this diagram, we may have many replica of Ga, but they correspond to different morphisms $Ja \to b$, and thus have different ingoing and outgoing morphisms. Finally, $\mathscr{E}G(b)$ is obtained as the limit of such diagram and, given $g : b \to b'$ in $\mathbf{B}$, $\mathscr{E}G(g) : \mathscr{E}G(b) \to \mathscr{E}G(b')$ is uniquely induced by the universal property of limits.

2.2. Coalgebras

The *behavior* of systems can be modeled in a categorical setting through *coalgebras* [11,20]. Given a *behavioral endofunctor* $B : \mathbf{C} \to \mathbf{C}$, describing the "shape" of a class of systems, we have a corresponding category of coalgebras.

Definition 2.4 *(*B**-Coalg**). The category B-**Coalg** is defined as follows: objects are *B-coalgebras*, i.e. pairs (X, h) of an object $X \in |\mathbf{C}|$, called *carrier*, and a morphism $h : X \to BX$, called *structure map*; *B-coalgebra homomorphisms* $f : (X, h) \to (Y, g)$ are morphisms $f : X \to Y$ in $\mathbf{C}$ making the following diagram commute

$$\begin{array}{ccc} X & \xrightarrow{h} & BX \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle Bf} \\ Y & \xrightarrow[g]{} & BY \end{array}$$

For instance, consider the functor

$$B_{clts} := \mathcal{P}_c(L \times -)$$

where $\mathcal{P}_c : \mathbf{Set} \to \mathbf{Set}$ is the *countable powerset functor*, given by

$$\mathcal{P}_c A := \{B \subseteq A \mid B \text{ countable}\} \qquad \mathcal{P}_c f(B) := \{f(b) \mid b \in B\} \qquad (f : A \to A', B \in \mathcal{P}_c A)$$

B_{clts}-coalgebras are *countably-branching labeled transition systems*, with labels in L, and their homomorphisms are functions that preserve and reflect transitions.

In this category many notions of behavioral equivalence can be defined [21]. We adopt the following one.

Definition 2.5 *(B-bisimulation).* Given a B-coalgebra (X, h), a *B-bisimulation* on it is an object R of $\mathbf{C}$ such that $R \hookrightarrow X \times X$ and there is $r : R \to BR$ making the following diagram commute

$$\begin{array}{ccccc} X & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & X \\ {\scriptstyle h}\downarrow & & {\scriptstyle r}\downarrow & & \downarrow{\scriptstyle h} \\ BX & \xleftarrow[B\pi_1]{} & BR & \xrightarrow[B\pi_2]{} & BX \end{array}$$

A B_{clts}-bisimulation R on a B_{clts}-coalgebra is an ordinary bisimulation on the corresponding transition system: the diagram means that $x, x' \in X$ such that $(x, x') \in R$ must be able to perform transitions with the same label l, and these transitions are represented as a single one $(x, x') \xrightarrow{l} (y, y')$ in (R, r); by definition of B_{clts}, y and y' must again be related by R.

An important property of categories of coalgebras is the existence of the terminal object; the unique morphism from each coalgebra to it assigns to each state its abstract semantics. If B preserves weak pullbacks, i.e. pullbacks such that the mediating morphism need not be unique, then B-bisimilarity and the terminal coalgebra agree.

The requirement for the existence of the final coalgebra is that B is an *accessible* functor on a *locally presentable* category (see [22,23,20] for details). We just recall the main results. Given a regular cardinal λ, a category $\mathbf{C}$ is *λ-filtered* if each diagram of cardinality less than λ is the base of a cocone in $\mathbf{C}$. λ-filtered categories generalize the notion of directed preorders, that are sets such that every finite subset has an upper bound. For any category $\mathbf{D}$, a *λ-filtered colimit* in $\mathbf{D}$ is the colimit of a diagram of shape $\mathbf{C}$, i.e. a functor $\mathbf{C} \to \mathbf{D}$, such that $\mathbf{C}$ is a λ-filtered category.

Definition 2.6 *(Locally λ-presentable category).* An object c of a category $\mathbf{C}$ is λ-presentable if the functor $\mathrm{Hom}_{\mathbf{C}}(c, -) : \mathbf{C} \to \mathbf{Set}$ preserves λ-filtered colimits. A category $\mathbf{C}$ is locally λ-presentable if it has all colimits and there is a set of λ-presentable objects $X \subseteq |\mathbf{C}|$ such that every object is a λ-filtered colimit of objects from X. We say that a category is locally presentable if it is locally λ-presentable for some λ.

For instance, locally λ-presentable objects in **Set** are precisely the finite sets with cardinality less than λ. **Set** is locally ω-presentable: every set is the ω-filtered colimit of its finite subsets and the whole **Set** is generated by the set containing one finite set of cardinality n for all $n \in \mathbb{N}$.

For functor categories we have the following.

Proposition 2.7. *For each locally λ-presentable category $\mathbf{C}$ and small category $\mathbf{D}$, the functor category $\mathbf{C}^{\mathbf{D}}$ is λ-presentable.*

In particular, since **Set** is ω-presentable, we have that the presheaf category $\mathbf{Set}^{\mathbf{D}}$ is ω-presentable as well.

Definition 2.8 *(Accessible functor).* Let $\mathbf{C}$ and $\mathbf{D}$ be locally λ-presentable categories. A functor $F : \mathbf{C} \to \mathbf{D}$ is *λ-accessible* if it preserves λ-filtered colimits. We just say F is accessible if it is λ-accessible for some λ.

Products, coproducts and composition of accessible functors are accessible as well.

2.3. Coalgebras over presheaves

Coalgebras for functors $B : \mathbf{Set}^{\mathbf{C}} \to \mathbf{Set}^{\mathbf{C}}$ have some additional structure: they are pairs (P, ρ) of a presheaf $P : \mathbf{C} \to \mathbf{Set}$ and a natural transformation $\rho : P \to BP$. The naturality of ρ imposes a constraint on behavior

$$\begin{array}{ccc} c & & p \in Pc \xmapsto{\rho_c} beh(p) \\ {\scriptstyle f}\downarrow & & {\scriptstyle [f]_P}\downarrow \qquad\qquad \downarrow{\scriptstyle [f]_{BP}} \\ c' & & p[f]_P \in P(c') \xmapsto[\rho_{c'}]{} beh(p)[\sigma]_{BP} \end{array}$$

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

$$
\begin{array}{rl}
p & ::= \mathbf{0} \mid \pi.p \mid p+p \mid p \mid p \mid (r)p \mid A(r_1, r_2, \ldots, r_n) \\
\pi & ::= \bar{a}r \mid a(r) \mid l_{ab} \mid \tau \\
r & ::= a \mid l_{ab} \\
\multicolumn{2}{c}{A(r_1, r_2, \ldots, r_n) \stackrel{\text{def}}{=} p \qquad i \neq j \Longrightarrow \mathrm{n}(r_i) \cap \mathrm{n}(r_j) = \varnothing}
\end{array}
$$

Fig. 2. NCPi syntax.

Intuitively, this diagram means that, if we take a state, apply a function to it and then compute its behavior, we should get the same thing as first computing the behavior and then applying the function to it. In other words, behavior must be *preserved* and *reflected* by the index category morphisms.

Also bisimulations have more structure. A B-bisimulation R is a presheaf in $\mathbf{Set}^{\mathbf{C}}$ and all the legs of the bisimulation diagram in Definition 2.5 are natural transformations. In particular, the naturality of projections implies that, given $(p, q) \in Rc$ and $f : c \to c'$ in $\mathbf{C}$, $(p[f], q[f]) \in R(c')$, i.e. B-bisimulations are *closed under the index category morphisms*.

In order to establish a correspondence between coalgebras over presheaves and transition systems, and between coalgebraic and ordinary bisimulations, in [24] transition systems and bisimulations over indexed states are introduced. The original definition is tailored for the π-calculus, but we give a more general one.

Definition 2.9 (**C**-*indexed labeled transition system*). Given a set of labels L, a **C**-*indexed labeled transition system* (**C**-ILTS) is a pair $(P, \longrightarrow)$ of a presheaf $P : \mathbf{C} \to \mathbf{Set}$ and a transition relation $\longrightarrow \subseteq \int P \times L \times \int P$.

Definition 2.10 (**C**-*indexed bisimulation*). A **C**-*indexed bisimulation* on a **C**-ILTS $(P, \longrightarrow)$ is an indexed family of relations $\{R_c \subseteq Pc \times Pc\}_{c \in |\mathbf{C}|}$ such that, for all $(p, q) \in R_c$

(i) if $c \vdash p \xrightarrow{l} c' \vdash p'$ then there is $c' \vdash q'$ such that $c \vdash q \xrightarrow{l} c' \vdash q'$ and $(p', q') \in R_{c'}$;
(ii) for all $f : c \to c'$, $(p[f], q[f]) \in R_{c'}$.

3. NCPi

3.1. Syntax

We assume an enumerable set of site names $\mathcal{S}$ (or just sites) and an enumerable set of link names $\mathcal{L}$ (or just links), equipped with two functions $s, t : \mathcal{L} \to \mathcal{S}$, telling source and target of each link. We denote by l_{ab} a link l such that $s(l) = a$ and $t(l) = b$. We write $\mathcal{L}_{ab}$ for the set of links of the form l_{ab} and $\mathcal{L}_a$ for the union of all $\mathcal{L}_{ab}$ and $\mathcal{L}_{ba}$, for all b.

The syntax of NCPi is given in Fig. 2: $\mathrm{n}(r)$ denotes the names in r, including a and b if r is l_{ab}. We have the usual inert process, sum and parallel composition. Prefixes can have the following forms:

- The *output prefix* $\bar{a}r$: $\bar{a}r.p$ can emit the datum r at a and continue as p.
- The *input prefix* $a(r)$: $a(r).p$ can receive at a a datum to be bound to r and becomes p.
- The *link prefix* l_{ab}: $l_{ab}.p$ can offer to the environment the service of transporting a datum from a to b through l and then continue as p.
- The τ *prefix*: $\tau.p$ can perform an internal action and continue as p.

We require that formal parameters in definitions do not have names in common, because otherwise we might have type dependencies between parameters, e.g. in $A(a, l_{ab})$ one of the second parameter's endpoints depends on the first parameter.

The free names $\mathrm{fn}(p)$ of a process p are

$$
\begin{array}{rlrl}
\mathrm{fn}(\mathbf{0}) & := \varnothing & \mathrm{fn}(\tau.p) & := \mathrm{fn}(p) \\
\mathrm{fn}(\bar{a}r.p) & := \{a\} \cup \mathrm{n}(r) \cup \mathrm{fn}(p) & \mathrm{fn}(l_{ab}.p) & := \{l_{ab}, a, b\} \cup \mathrm{fn}(p) \\
\mathrm{fn}(b(a).p) & := \{b\} \cup (\mathrm{fn}(p) \setminus (\{a\} \cup \mathcal{L}_a)) & \mathrm{fn}(a(l_{bc}).p) & := \{a, b, c\} \cup \mathrm{fn}(p) \setminus \{l_{bc}\} \\
\mathrm{fn}((a)p) & := \mathrm{fn}(p) \setminus (\{a\} \cup \mathcal{L}_a) & \mathrm{fn}((l_{ab})p) & := \{a, b\} \cup \mathrm{fn}(p) \setminus \{l_{ab}\} \\
\mathrm{fn}(p+q) & := \mathrm{fn}(p \mid q) := \mathrm{fn}(p) \cup \mathrm{fn}(q) & \mathrm{fn}(A(r_1, \ldots, r_n)) & := \mathrm{n}(r_1) \cup \cdots \cup \mathrm{n}(r_n)
\end{array}
$$

where we must have $\mathrm{fn}(p) \subseteq \mathrm{fn}(A(r_1, \ldots, r_n))$ whenever $A(r_1, r_2, \ldots, r_n) \stackrel{\text{def}}{=} p$. Notice the cases $(a)p$ and $b(a).p$: a free link in p having a as endpoint is considered bound in $(a)p$ and $b(a).p$. This intuitively means that a global link cannot have private endpoints. Given a name r, we shall write $r \# p$ to indicate that r is *fresh* w.r.t. p, i.e. $r \notin \mathrm{fn}(p)$; $N \# p$, with N a set of names, has the expected meaning.

Now we define renamings. In the following, we shall use the usual notation $[r'_1/r_1, r'_2/r_2, \ldots, r'_n/r_n]$ to indicate the substitution mapping r_1 to r'_1, r_2 to $r'_2 \ldots r_n$ to r'_n. In our case, we call *renaming* a substitution that is "well-behaved" with respect to the graph-structure described by names. In other words, it should be a *graph homomorphism*, i.e. each link should be mapped to one whose endpoints are the image through the substitution of the original link's endpoints.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

α-equivalence

$$\begin{array}{lll} (a)p \equiv (a')p[a'/a] & b(a).p \equiv b(a').p[a'/a] & a' \# (a)p \\ (l_{ab})p \equiv (l'_{ab})p[l'_{ab}/l_{ab}] & c(l_{ab}).p \equiv c(l'_{ab}).p[l'_{ab}/l_{ab}] & l'_{ab} \# (l_{ab})p \end{array}$$

Commutative monoid laws for $|$ and $+$

$$p_1 \mid p_2 \equiv p_2 \mid p_1 \quad p_1 \mid (p_2 \mid p_3) \equiv (p_1 \mid p_2) \mid p_3 \quad p \mid \mathbf{0} \equiv p \quad \text{(similarly for } +)$$

Scope extension laws

$$\begin{array}{ll} p_1 \mid (r)p_2 \equiv (r)(p_1 \mid p_2) & r \# p_1 \\ (r)(r')p \equiv (r')(r)p & r \notin \mathrm{n}(r') \end{array}$$

Unfolding law

$$A(r'_1, \ldots, r'_n) \equiv p[\mathrm{n}(r'_1)/\mathrm{n}(r_1), \ldots, \mathrm{n}(r'_n)/\mathrm{n}(r_n)] \quad \text{if } A(r_1, \ldots, r_n) \stackrel{\text{def}}{=} p$$

Fig. 3. Structural congruence axioms for well-formed processes.

$$\begin{array}{rcll} \alpha & ::= & a; W; b \mid \bullet; W; \bullet \mid ar; \bullet \mid \bullet; W; \bar{a}r \mid \bullet; W; \bar{a}(r') & r' \notin \mathrm{n}(W) \cup \{a\} \\ r, r' & ::= & a \mid l_{ab} & \\ W & ::= & l_{ab} \mid W; W \mid \varepsilon & \end{array}$$

structural congruence $\equiv_\alpha$ is given by the monoidal axioms for strings, where ; is the multiplication and ε the identity

Fig. 4. Syntax of paths.

Definition 3.1 *(Renaming).* A renaming σ is a pair of functions $\langle \sigma_{\mathcal{S}} : \mathcal{S} \to \mathcal{S}, \sigma_{\mathcal{L}} : \mathcal{L} \to \mathcal{L} \rangle$ such that

$$\sigma_{\mathcal{L}}(l_{ab}) = l'_{a'b'} \Longrightarrow \sigma_{\mathcal{S}}(a) = a' \wedge \sigma_{\mathcal{S}}(b) = b'.$$

In order to define the extension of renamings to processes, we need a notion of α-conversion that establishes how to avoid captures. Such notion will rely on substitutions of the form $[a'/a]$ in order to α-convert w.r.t. sites. However, such a substitution does not uniquely characterize a renaming. In fact, while surely $\bar{a}c[a'/a] = \overline{a'}c$, for $l_{ab}[a'/a]$ we only know that it must belong to $\mathcal{L}_{a'b}$. If, however, l_{ab} is bound, the choice of $l_{ab}[a'/a]$ is immaterial, provided that we remain in the same α-equivalence class. This motivates the introduction of *well-formed* processes. Informally, a process is well-formed if, in any of its subprocesses, bound links are bound explicitly, and not as a side-effect of binding a site. For instance, $a(b).l_{bc}.p$ is not well-formed because l_{bc} is implicitly bound by $a(b)$.

Definition 3.2 *(Well-formed process).* A NCPi process p is *well-formed* if for every subterm q:

(i) $q = (a)p'$ implies $\mathrm{fn}(q) = \mathrm{fn}(p') \setminus \{a\}$;
(ii) $q = b(a).p'$ implies $\mathrm{fn}(q) = \{b\} \cup \mathrm{fn}(p') \setminus \{a\}$;

Notice that, as a consequence of this definition, we do not need to subtract $\mathcal{L}_a$ when computing the free names of $b(a).p$ and $(a)p$, if these processes are well-formed. In the following we assume that processes are always well-formed. This allows us to define the extension of renamings to processes, which is in some sense "mutually recursive" with α-conversion.

Definition 3.3 *(Process renamings).* Given a renaming σ and a well-formed process p, we denote by $p\sigma$ the result of applying σ to $\mathrm{fn}(p)$ with α-conversion of bound names to avoid captures of sites and links.

Finally, structural congruence axioms for well-formed processes are listed in Fig. 3. Here we write $\mathrm{n}(r')/\mathrm{n}(r)$ for the substitutions $a'/a, b'/b, l'_{a'b'}/l_{ab}$ (resp. a'/a) whenever $r = l_{ab}$ and $r' = l'_{a'b'}$ (resp. $r = a$ and $r' = a'$). The interesting case is α-conversion w.r.t. a site: when α-converting $(a)p$, $[a'/a]$ is never applied to a link l_{ab}, since such link cannot be free in p by well-formedness; if it is bound, i.e. if $(l_{ab})p'$ is a subprocess of p, then we simply have inductively $((l_{ab})p')[a'/a] \equiv (l'_{a'b})p'[l'_{a'b}/l_{ab}][a'/a]$, for any $l'_{a'b}$ fresh w.r.t. p. The axioms' side conditions guarantee preservation of well-formedness.

3.2. Semantics

Observations of the operational semantics are routing paths, whose syntax is given in Fig. 4.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

Table 1
Free names, bound names, objects, input objects and interaction sites of a path α.

path α	fn	bn	obj	obj_{in}	is
$a; W; b$	$\text{n}(\alpha)$	$\varnothing$	$\varnothing$	$\varnothing$	$\{a, b\}$
$\bullet; W; \bullet$	$\text{n}(\alpha)$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
$\bullet; W; \bar{a}r$	$\text{n}(\alpha)$	$\varnothing$	$\text{n}(r)$	$\varnothing$	$\{a\}$
$\bullet; W; a(r)$	$\text{n}(\alpha) \setminus \{r\}$	$\{r\}$	$\text{n}(r) \setminus \{r\}$	$\varnothing$	$\{a\}$
$ar; \bullet$	$\text{n}(\alpha)$	$\varnothing$	$\text{n}(r)$	$\text{n}(r)$	$\{a\}$

A path α can be of two general forms. It can be a *service path* $a; W; b$, representing a transportation service from a to b that employs the links listed in W and possibly other private, unobservable ones. Alternatively, it can be a string starting and/or ending with $\bullet$, which represents an actual transmission. More specifically, in this case α can be:

- an *output path*, if $\bar{a}r$ or $\bar{a}(r)$ occurs on the right, after a (possibly empty) sequence of links W: in both cases α represents r being emitted at a after traveling along W; if r is free then α is called *free output path*, otherwise α is called *bound output path* and represents an extrusion.
- an *input path*, if ar is on the left, representing the reception of r at a.
- a *complete path*, if $\bullet$ is on both sides of W, meaning that a transmission over the links in W has been completed.

Notice that input and output paths are not symmetrical: only output paths exhibit a list W of employed links. This is mainly for simplicity of presentation, and follows the intuition that a datum travels from the sender to its destination.

We call *interaction sites* of α, written $\text{is}(\alpha)$ and defined in Table 1, those sites where the interaction with another process may happen. These correspond to subjects of the π-calculus. Table 1 also defines the free names $\text{fn}()$, bound names $\text{bn}()$, objects $\text{obj}()$, input objects $\text{obj}_{\text{in}}()$ of α. Given a list of links W, we write W/r for W after removing each occurrence of $r \in \mathcal{L}$, and α/r for α with $/r$ applied to its list of links.

Definition 3.4 *(NCPi transition system).* The *NCPi transition system* is the smallest transition system generated from the rules in Fig. 5, where observations are up to $\equiv_\alpha$ and transitions are closed under $\equiv$, i.e. if $p \xrightarrow{\alpha} q$, $p \equiv p'$, $q \equiv q'$ and $\alpha' \equiv_\alpha \alpha$ then $p' \xrightarrow{\alpha'} q'$.

Now we briefly explain the rules. (OUT) and (IN) infer a zero-length path representing, respectively, the beginning and the end of a transmission. As in the early π-calculus, a renaming must be applied to the continuation in the free input case; if the input object is a site a, then we have a substitution between sites, which can be turned into a proper renaming by well-formedness. (LINK) infers a service path made of one link. (INT) infers an internal action, represented as a complete path where everything is unobservable. (RES) computes the paths of a process with an additional restriction (r) from those of the unrestricted process, provided that r is not already bound and is not an object or an interaction site. This side condition reflects that of the corresponding π-calculus rule, where r must not be the subject or the object of the premise's action, and its purpose is to avoid captures. In fact, suppose $b(c).p$ can perform $ba; \bullet$ and $c \in \text{fn}(p)$: if $(a)b(c).p$ is allowed to perform the same path then a would be captured in its continuation $(a)p[a/c]$. As a result of the restriction, r becomes unobservable, so it is removed from the list of employed links in the inferred path. (OPEN) treats the case, excluded by (RES), when r is the object of a free output path: such path is turned into a bound output path, again rendering r unobservable when needed. (SUM) and (PAR) are as expected. (ROUTE), (COMPOSE) and (COM) concatenate paths that meet at an interaction site: (ROUTE) extends an output path, provided that the transported name, whenever bound, is fresh w.r.t. the process that offers the transportation service; (COMPOSE) composes two service paths; (COM) completes a communication.

Remark 3.5. For the sake of symmetry, we could have input paths that include a W component, like output paths, listing the links that a datum can traverse in order to reach the site where an input is performed. Moreover, we could have an additional inference rule, dual to (ROUTE), for adding links to such paths. However, this would not yield additional observations. In fact, any complete path can be derived starting with (OUT), then using (ROUTE) to attach transportation services provided by parallel processes and finally (COM) to finalize the communication.

We have that the transition system generated by these rules behaves well w.r.t. well-formedness.

Proposition 3.6. *If p is well-formed and $p \xrightarrow{\alpha} q$ then q is well-formed.*

The notion of behavioral equivalence is the following one, called *network conscious bisimulation*.

Definition 3.7 *(Network conscious bisimulation).* A binary, symmetric and reflexive relation R is a *network conscious bisimulation* if $(p, q) \in R$ and $p \xrightarrow{\alpha} p'$, with $\text{bn}(\alpha) \# q$, implies that there is q' such that $q \xrightarrow{\alpha} q'$ and $(p', q') \in R$. The bisimilarity is the largest such relation and is denoted by $\sim^{NC}$.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

$$\text{(OUT)}\quad \bar{a}r.p \xrightarrow{\bullet;\bar{a}r} p \qquad \text{(OPEN)}\quad \frac{p \xrightarrow{\bullet;W;\bar{a}r} q}{(r)p \xrightarrow{\bullet;W/r;\bar{a}(r)} q}\ r\neq a$$

$$\text{(IN)}\quad a(r).p \xrightarrow{ar';\bullet} p[r'/r] \qquad \text{(PAR)}\quad \frac{p_1 \xrightarrow{\alpha} q_1}{p_1 \mid p_2 \xrightarrow{\alpha} q_1 \mid p_2}\ \mathrm{bn}(\alpha)\,\#\,p_2$$

$$\text{(LINK)}\quad l_{ab}.p \xrightarrow{a;l_{ab};b} p \qquad \text{(ROUTE)}\quad \frac{p_1 \xrightarrow{\bullet;W;\bar{a}x} q_1 \qquad p_2 \xrightarrow{a;W;b} q_2}{p_1 \mid p_2 \xrightarrow{\bullet;W;W';\bar{b}x} q_1 \mid q_2}\ \mathrm{bn}(x)\,\#\,p_2$$

$$\text{(INT)}\quad \tau.p \xrightarrow{\bullet;\bullet} p \qquad \text{(COMPOSE)}\quad \frac{p_1 \xrightarrow{a;W;b} q_1 \qquad p_2 \xrightarrow{b;W';c} q_2}{p_1 \mid p_2 \xrightarrow{a;W;W';c} q_1 \mid q_2}$$

$$\text{(SUM)}\quad \frac{p \xrightarrow{\alpha} p'}{p+q \xrightarrow{\alpha} p'} \qquad \text{(COM)}\quad \frac{p_1 \xrightarrow{\bullet;W;\bar{a}r} q_1 \qquad p_2 \xrightarrow{ar;\bullet} q_2}{p_1 \mid p_2 \xrightarrow{\bullet;W;\bullet} q_1 \mid q_2}$$

$$\text{(RES)}\quad \frac{p \xrightarrow{\alpha} q}{(r)p \xrightarrow{\alpha/r} (r)q}\ r\notin \mathrm{bn}(\alpha)\cup\mathrm{obj}(\alpha)\cup\mathrm{is}(\alpha)$$

Fig. 5. NCPi SOS rules.

Notice that a consequence of defining the semantics up to structural congruence is that $\equiv \subseteq \sim^{NC}$.
It is easy to see that the π-calculus is included in NCPi.

Definition 3.8 *(Linkless NCPi).* We call *linkless NCPi* (NCPi$_{-\ell}$) the subcalculus of NCPi such that no links appear in processes.

Clearly, NCPi$_{-\ell}$ processes are π-calculus processes. The induced restriction on SOS rules in Fig. 5, together with the following encoding of π-calculus actions

$$\begin{aligned} \bar{a}x &\longmapsto \bullet;\bar{a}x \quad x\in\{b,(b)\} \\ ab &\longmapsto ab;\bullet \\ \tau &\longmapsto \bullet;\bullet \end{aligned}$$

give the following.

Proposition 3.9. *NCPi$_{-\ell}$ transitions are in bijective correspondence with π-calculus early transitions.*

3.3. Closure properties

Here we list some properties of the transition system and its bisimulations, namely closure under some classes of renamings and contexts. Their proofs are standard.

We say that a renaming σ is *injective* if so are $\sigma_{\mathcal{S}}$ and $\sigma_{\mathcal{L}}$. We have that the transition system is closed under injective renamings.

Proposition 3.10. *Let p be a process and σ an injective renaming, then:*

(i) *if $p \xrightarrow{\alpha} p'$ then $p\sigma \xrightarrow{\alpha\sigma} p'\sigma$ (transitions are preserved by σ);*
(ii) *if $p\sigma \xrightarrow{\alpha} p'$ then there is $p \xrightarrow{\alpha'} p''$ such that $\alpha'\sigma \equiv_\alpha \alpha$ and $p''\sigma \equiv p'$ (transition are reflected by σ).*

Remark 3.11. Transitions are *not* reflected by generic renamings. In fact, consider the process $p \stackrel{\text{def}}{=} \bar{a}r.\mathbf{0} \mid l_{ab}.\mathbf{0} \mid b'(x).\mathbf{0}$ and the renaming σ that maps b' to b and acts as the identity elsewhere. Then we have $p\sigma \xrightarrow{\bullet;l_{ab};\bullet} \mathbf{0}$ but there is no α' such that $p \xrightarrow{\alpha'} \mathbf{0}$ and $\alpha'\sigma \equiv_\alpha \bullet;l_{ab};\bullet$.

Proposition 3.12. *$\sim^{NC}$ is closed under injective renamings, i.e. $p \sim^{NC} q$ implies $p\sigma \sim^{NC} q\sigma$, for all injective σ.*

Theorem 3.13. *$\sim^{NC}$ is closed under all operators except the input prefix and parallel composition.*

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

Example 3.14. Consider the two processes

$$p \stackrel{\text{def}}{=} (l_{cb})l_{cb}.\mathbf{0} \mid l'_{b'd}.\mathbf{0} \qquad q \stackrel{\text{def}}{=} (l_{cb})l_{cb}.l'_{b'd}.\mathbf{0} + l'_{b'd}.(l_{cb})l_{cb}.\mathbf{0}$$

We have $p \sim^{NC} q$ but

$$a(b).p \xrightarrow{ab';\bullet} (l_{cb'})l_{cb'}.\mathbf{0} \mid l'_{b'd}.\mathbf{0} \xrightarrow{c;l'_{b'd};d} \mathbf{0}$$
$$a(b).q \xrightarrow{ab';\bullet} (l_{cb'})l_{cb'}.l'_{b'd}.\mathbf{0} + l'_{b'd}.(l_{cb'})l_{cb'}.\mathbf{0} \not\xrightarrow{c;l'_{b'd};d}$$

which implies $a(b).p \not\sim^{NC} a(b).q$.

Theorem 3.15. *The greatest network conscious bisimulation that is closed under all renamings is a congruence for all operators except parallel composition. Closure under this operator is discussed in the note added in the proof.*

4. Coalgebraic semantics of NCPi

In this section we characterize the NCPi syntax and semantics in a categorical context. Our main reference is [10], which concerns the π-calculus, but we will suitably adapt it for our calculus. We start with an overview of the approach.

4.1. The approach

The high-level steps of the approach are:

(i) select a category $\mathbf{C}$ of indices representing resources and their operations, together with endofunctors $\delta : \mathbf{C} \to \mathbf{C}$ that model resource generation;
(ii) model processes and renamings as a syntactic presheaf in $\mathbf{Set}^{\mathbf{C}}$, freely generated from a signature endofunctor which employs each δ to represent resource allocation primitives;
(iii) model the transition system as a coalgebra over the syntactic functor, for a behavioral endofunctor where each δ is used to represent a way of allocating resources along transitions.

For instance, in the case of the π-calculus, $\mathbf{C}$ is the category $\mathbf{F}$ of finite ordinals and functions, representing finite sets of names and renamings. It is equipped with an endofunctor $\delta : \mathbf{F} \to \mathbf{F}$ that adds a fresh name to its argument. The syntactic functor maps each set of names to processes where (at most) such names occur free, and it is computed as the initial algebra for a signature endofunctor where δ is used to model binding operators, namely restriction and input prefix.

However, there is a difficulty in step (iii): while syntax can be modeled in $\mathbf{Set}^{\mathbf{F}}$, the transition system cannot, because transitions are not reflected by all renamings, and neither can its bisimilarity, because it is not closed under the input prefix or, equivalently, under all renamings. The solution is splitting step (iii) in two substeps:

(iii.a) identify a subcategory $\mathbf{C}'$ of $\mathbf{C}$ such that the transition system is closed under its morphisms, and construct a coalgebra in $\mathbf{Set}^{\mathbf{C}'}$ by suitably restricting the syntactic presheaf;
(iii.b) recover a coalgebra in $\mathbf{Set}^{\mathbf{C}}$ via *right Kan extension* (see Section 2.1 and [19]) along the embedding $\mathbf{C}' \hookrightarrow \mathbf{C}$.

The category $\mathbf{C}'$, for the π-calculus, is $\mathbf{I}$, the subcategory of $\mathbf{F}$ with only injections. A faithful coalgebraic representation of the π-calculus transition and of (early/late) observational equivalence is then feasible, because they are known to be closed under injective renamings. Step (iii.b) is accomplished by enriching the behavioral functor of step (iii.a), so that each transitions also expresses the application of a renaming. This is a form of *saturation* [25], as shown in [26]. In the resulting category of coalgebras the greatest bisimulation characterizes observational (early/late) congruence, because behavior is always closed under all renamings. In our case, we do not get a full congruence (see note added in proof).

Besides the steps of [10], we will also consider the construction of a finite-state representation of our coalgebraic semantics, in the form of a HD-automaton. Such semantics, in fact, will have an infinite number of states, due to lack of deallocation along transitions. For the π-calculus, this has been done by exploiting the equivalence between coalgebras on pullback-preserving presheaves in $\mathbf{Set}^{\mathbf{I}}$ and HD-automata [27,24]. A recent generalization [12] characterizes a spectrum of presheaf categories that admit HD-automata. We will employ this result to show the existence of a HD-automaton corresponding to the NCPi coalgebra with ordinary bisimulation. Unfortunately, as in the π-calculus case, the category of saturated coalgebras is based on presheaves that are not covered by the result of [12].

4.2. Categorical environment

Resources of NCPi processes, namely communication networks, can be formally seen as finite, directed multigraphs, so we define a category made of this kind of graphs and their homomorphisms. We adopt the presentation of such graphs as

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

functors from the category $\rightrightarrows$ with two objects, representing vertices and edges, and two parallel morphisms, representing source and target operations, to the category of finite sets and functions **FinSet**. However, we just take a skeletal category of $\mathbf{FinSet}^{\rightrightarrows}$, analogously to what Fiore and Turi do for finite sets in [10].

Definition 4.1 *(Category* **G***).* We denote by **G** the skeletal category of $\mathbf{FinSet}^{\rightrightarrows}$.

We don't give an explicit construction for **G**: all choices are consistent, since they are all isomorphic. This is why we refer to **G** as "the" skeletal category.

Concretely, we can regard each $g \in |\mathbf{G}|$ as a tuple (v_g, e_g, s_g, t_g), where v_g, e_g are the sets of vertices and edges of g, and $s_g, t_g : e_g \to v_g$ tell the source $s_g(e)$ and target $t_g(e)$ of each $e \in e_g$. A morphism $\sigma : g \to g'$ is a natural transformation, i.e. a pair of functions (σ_v, σ_e) that commute with the source and target functions of g and g', which is exactly the definition of graph homomorphism. We state some properties of **G** that will be important in the following.

Proposition 4.2. *The category* **G** *is small, has finite colimits and pullbacks.*

$\mathbf{FinSet}^{\rightrightarrows}$ is locally small, but not small: this is why we consider a skeletal version of it. Some notation: we write $[n]$ for the discrete graph with n vertices, and k_n for the graph with n vertices and with one edge between every (ordered) pair of vertices.

Thanks to Proposition 4.2, we can exploit colimits to implement allocation of fresh resources.

Allocation of vertices Given $g \in |\mathbf{G}|$, we can express the allocation of a fresh, disconnected vertex $\star$ as a coproduct

$$[1] \xrightarrow{new_g^\bullet} g + [1] \xleftarrow{old_g^\bullet} g$$

This induces the endofunctor $\delta^\bullet : \mathbf{G} \to \mathbf{G}$ given by

$$\delta^\bullet g := g + [1] \qquad \delta^\bullet \sigma := \sigma + id_{[1]}.$$

Allocation of edges Given $g \in |\mathbf{G}|$ with n vertices, we can add a new edge $\star_{ij}$ between each ordered pair of vertices i and j through a pushout

$$\begin{array}{ccc} [n] & \hookrightarrow & g \\ \downarrow & \lrcorner & \downarrow{\scriptstyle old_g^{\bullet\to\bullet}} \\ k_n & \xrightarrow[new_g^{\bullet\to\bullet}]{} & g^* \end{array}$$

that makes the disjoint union of the items of k_n and g, and then identifies the vertices that are image of the same vertex in $[n]$ through the embeddings. Given g_1 and g_2 in **G**, with n_1 and n_2 vertices respectively, every $\sigma : g_1 \to g_2$ in **G** can be canonically extended to a morphism $\sigma^* : g_1^* \to g_2^*$ via the universal property of pushouts as follows

$$\begin{array}{ccccc} [n_1] & \hookrightarrow & g_1 & \xrightarrow{\sigma} & g_2 \\ \downarrow & & \downarrow & & \\ k_{n_1} & \longrightarrow & g_1^* & \overset{\sigma^*}{\cdots\!\searrow} & \downarrow \\ \downarrow{\scriptstyle \widehat{\sigma}} & & & & \\ k_{n_2} & & \longrightarrow & & g_2^* \end{array}$$

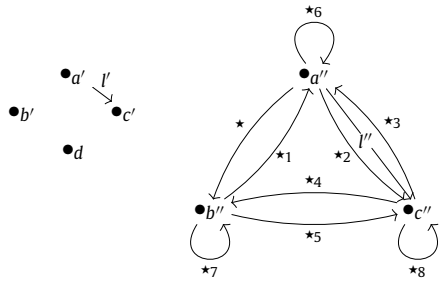
where $\widehat{\sigma}$ is the (unique) morphism between k_{n_1} and k_{n_2} that acts on vertices like σ (the action on edges is obvious). This construction can be turned into a functor $\delta^{\bullet\to\bullet} : \mathbf{G} \to \mathbf{G}$

$$\delta^{\bullet\to\bullet} g := g^* \qquad \delta^{\bullet\to\bullet} \sigma := \sigma^*.$$

Example 4.3. Consider the following graph

$$g := \quad \begin{array}{ccc} & \bullet_a & \\ & {\scriptstyle l}\searrow & \\ \bullet_b & & \bullet_c \end{array}$$

$\delta^\bullet g$ and $\delta^{\bullet\to\bullet} g$ are (isomorphic to)



where a', b', c' and l' (resp. a'', b'', c'', l'') are the images of a, b, c and l via $old^\bullet$ (resp. $old^{\bullet\to\bullet}$), d is the image of $\bullet$ via $new^\bullet$ and $\star, \star_1, \ldots, \star_8$ are the images of the edges in k_3 via $new^{\bullet\to\bullet}$.

We denote by $\mathbf{G}_\mathbf{I}$ the subcategory of $\mathbf{G}$ with only monos. We remark that $\mathbf{G}_\mathbf{I}$ lacks pushouts, but we can compute them in $\mathbf{G}$, since monos are stable under pushouts in $\mathbf{G}$. Consequently, $\delta^\bullet$ and $\delta^{\bullet\to\bullet}$ are well-defined also in $\mathbf{G}_\mathbf{I}$.

Now we look at the category $\mathbf{Set}^\mathbf{G}$ of presheaves on graphs. As mentioned, it is locally presentable and has all limits and colimits, in particular products and coproducts. The following constructs are relevant for us.

Name functors $\mathcal{S}, \mathcal{L} : \mathbf{G} \to \mathbf{Set}$ giving, for each $g \in |\mathbf{G}|$, the set of sites and links corresponding to the vertices and edges of g. Formally, let $\bullet$ be the graph with one vertex and no edges, and $\bullet \to \bullet$ be the graph with two vertices and an edge between then. We define

$$\mathcal{S} := \mathrm{Hom}_\mathbf{G}(\bullet, -) \qquad \mathcal{L} := \mathrm{Hom}_\mathbf{G}(\bullet \to \bullet, -) \qquad \mathcal{N} := \mathcal{S} + \mathcal{L}$$

Explicitly, $\mathcal{S}$ sends $g \in |\mathbf{G}|$ to $\mathbf{G}[\bullet, g]$, which is isomorphic to the set of vertices of g, and $\sigma : g \to g'$ in $\mathbf{G}$ to the function $\lambda s \in \mathrm{Hom}_\mathbf{G}(\bullet, g).\sigma \circ s$, which renames the site s according to σ; similarly for $\mathcal{L}$. In order to keep the same notation for names given in Section 3, given an edge l in g with endpoints a, b, the homomorphism $\bullet \to g$ in $\mathcal{S}g$ that maps $\bullet$ to a will be simply denoted by a; and the homomorphism $(\bullet \to \bullet) \to g$ in $\mathcal{L}g$ that maps the edge in the domain to l, and consequently its endpoints to a and b, will be denoted by l_{ab}.

Countable powerset $\mathscr{P}_c : \mathbf{Set}^\mathbf{G} \to \mathbf{Set}^\mathbf{G}$, defined as $\mathcal{P}_c \circ (-)$.

Allocation functors $\Delta^\bullet, \Delta^{\bullet\to\bullet} : \mathbf{Set}^\mathbf{G} \to \mathbf{Set}^\mathbf{G}$, defined as $(-) \circ \delta^\bullet$ and $(-) \circ \delta^{\bullet\to\bullet}$, respectively.

4.3. Coalgebraic semantics

Our aim now is to construct a coalgebra that models the NCPi transition system. Its carrier will be a suitable presheaf modeling processes and renamings. However, since transitions are not reflected by generic renamings (Remark 3.11), but only by injective ones (Proposition 3.10), according to step (iii.a) of Section 4.1 we first give a semantics in $\mathbf{Set}^{\mathbf{G}_\mathbf{I}}$.

Definition 4.4 *(Behavioral endofunctor).* Let $\mathcal{L}^\star = \sum_{i\in\omega} \mathcal{L}^i$. The behavioral endofunctor $B : \mathbf{Set}^{\mathbf{G}_\mathbf{I}} \to \mathbf{Set}^{\mathbf{G}_\mathbf{I}}$ is

$$\begin{aligned}
BP = \; & \mathscr{P}_c(\mathcal{S} \times \mathcal{L}^\star \times \mathcal{S} \times P && \text{(Service Path)}\\
& + \; \mathcal{L}^\star \times P && \text{(Complete Path)}\\
& + \; \mathcal{S} \times \mathcal{N} \times P && \text{(Known Name Input Path)}\\
& + \; \mathcal{S} \times \Delta^\bullet P && \text{(Fresh Site Input Path)}\\
& + \; \mathcal{S} \times \mathcal{S} \times \mathcal{S} \times \Delta^{\bullet\to\bullet} P && \text{(Fresh Link Input Path)}\\
& + \; \mathcal{L}^\star \times \mathcal{S} \times \mathcal{N} \times P && \text{(Free Output Path)}\\
& + \; \mathcal{L}^\star \times \mathcal{S} \times \Delta^\bullet P && \text{(Bound Site Output Path)}\\
& + \; \mathcal{L}^\star \times \mathcal{S} \times \mathcal{S} \times \mathcal{S} \times \Delta^{\bullet\to\bullet} P) && \text{(Bound Link Output Path)}
\end{aligned}$$

To understand this definition, consider a B-coalgebra (P, ρ). Given $g \in \mathbf{G}_\mathbf{I}$ and $p \in Pg$, $\rho_g(p)$ is a countable set of tuples. These tuples can be seen as pairs (α, p') of a path α and of a continuation from p after observing α, both built using the names corresponding to the items of g and possibly some fresh ones. We use the countable powerset because p might have recursive subprocesses that generate a countable number of looping paths. This does not affect the formal properties of B-coalgebras.

Notice the bound output cases: the continuation is drawn from $\Delta^\bullet P(g)$ or $\Delta^{\bullet\to\bullet} P(g)$, i.e. its index is $\delta^\bullet g$ or $\delta^{\bullet\to\bullet} g$; the extruded name, which corresponds to the new vertex or one of the new edges added to g by these functors, does not appear in α, because its identity is known. In the bound link output case the endpoints of the extruded link must be included in α, in order to allow processes that extrude links with different endpoints through the same path to be distinguished. The W component in Definition 4 is modeled through the functor $\mathcal{L}^*$, which returns the set of finite strings on the alphabet $\mathcal{L}g$.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

Input transitions are modeled similarly to free and bound output ones, even if there is no explicit binding: we distinguish between the reception of a known name, i.e. a name already in $\mathcal{N}g$, and of a fresh one; in the latter case, the index is augmented. This allows us to give a finite representation of an infinite number of transitions.

Theorem 4.5. *B is accessible and preserves weak pullbacks.*

In order to establish a correspondence between the set-theoretical notions of Section 3 and the world of coalgebras, we show that B-coalgebras can be represented as particular indexed labeled transition systems.

Definition 4.6 *(Network conscious $\mathbf{G}_\mathbf{I}$-ILTS).* Given the following presheaf of labels

$$Lab := \mathcal{S} \times \mathcal{L}^\star \times \mathcal{S} + \mathcal{L}^\star + \mathcal{S} \times \mathcal{N} + \mathcal{S} + \mathcal{S} \times \mathcal{S} \times \mathcal{S} + \mathcal{L}^\star \times \mathcal{S} \times \mathcal{N} + \mathcal{L}^\star \times \mathcal{S} + \mathcal{L}^\star \times \mathcal{S} \times \mathcal{S} \times \mathcal{S}$$

a *network conscious* $\mathbf{G}_\mathbf{I}$-ILTS ($\mathbf{G}_\mathbf{I}$-IL$_{nc}$TS) is a $\mathbf{G}_\mathbf{I}$-ILTS $(P, \longrightarrow)$ with labels in $\int Lab$, such that:

(i) Transitions have the following forms (the indices of labels, being the same as those of the source processes, are omitted):
- $g \vdash p \xrightarrow{\alpha} \delta^\bullet g \vdash p'$, with $\alpha = \bullet; W; \bar{a}(\star)$ (bound site output) or $\alpha = a\star; \bullet$ (fresh site input);
- $g \vdash p \xrightarrow{\alpha} \delta^{\bullet\to\bullet} g \vdash p'$, with $\alpha = \bullet; W; \bar{a}(\star_{bc})$ (bound link output) or $\alpha = a\star_{bc}; \bullet$ (fresh link input);
- $g \vdash p \xrightarrow{\alpha} g \vdash p'$ for all the other $\alpha \in Lab(g)$.

(ii) For each morphism $\sigma : g \to g'$ in $\mathbf{G}_\mathbf{I}$ and each $\varphi \in \{Id, \delta^\bullet, \delta^{\bullet\to\bullet}\}$: $g \vdash p \xrightarrow{\alpha} \varphi g \vdash p'$ if and only if $g' \vdash p[\sigma] \xrightarrow{\alpha[\sigma]} \varphi(g') \vdash p'[\varphi\sigma]$ (transition are preserved and reflected by morphisms).

Proposition 4.7. *B-coalgebras are in bijective correspondence with $\mathbf{G}_\mathbf{I}$-IL$_{nc}$TSs.*

For behavioral equivalences on $\mathbf{G}_\mathbf{I}$-IL$_{nc}$TSs, namely $\mathbf{G}_\mathbf{I}$-indexed bisimulations (Definition 2.9 instantiated with $\mathbf{C} = \mathbf{G}_\mathbf{I}$), we have the following correspondence.

Proposition 4.8. *Let (P, ρ) be a B-coalgebra. Then every B-bisimulation is equivalent to a $\mathbf{G}_\mathbf{I}$-indexed bisimulation on the induced $\mathbf{G}_\mathbf{I}$-IL$_{nc}$TS.*

Unfortunately, the converse is not true: there are bisimulations on some $\mathbf{G}_\mathbf{I}$-IL$_{nc}$TSs that cannot be turned into B-bisimulations. This has been pointed out in [24, 3.3.3, Anomaly] for the case of the π-calculus. The solution given there is to narrow the class of presheaves under consideration to *sheaves* in the *Schanuel topos*, that are sheaves $\mathbf{I} \to \mathbf{Set}$ for the coverage made of singleton families. These are exactly pullback-preserving presheaves [28, A.2, Example 2.1.1(h)]. Analogously, we can consider sheaves $\mathbf{G}_\mathbf{I} \to \mathbf{Set}$ for the same kind of coverage, since $\mathbf{G}_\mathbf{I}$ has pullbacks, and these again are presheaves that preserve pullbacks. As shown in [24, 4.2.1], for such sheaves we have notions such as *minimal supports* and *seeds*: given a sheaf P, $p \in Pg$ and $g' \hookrightarrow g$, there is a unique seed $p' \in P(g')$ such that $P(g' \hookrightarrow g)(p') = p$; moreover, there exists the smallest such graph g' where a seed of p can be found. In other words, we can always recover the minimal network a process uses. We recall Theorem 4.2.5 of [24], adapted to our context: it's easy to see that $\mathbf{G}_\mathbf{I}$-IL$_{nc}$TSs satisfy the relevant axioms characterizing the class of $\mathbf{I}$-ILTSs treated in [24].

Theorem 4.9. *Let (P, ρ) a B-coalgebra. If P is a sheaf then every $\mathbf{G}_\mathbf{I}$-indexed bisimulation on the induced $\mathbf{G}_\mathbf{I}$-IL$_{nc}$TS is also a B-bisimulation on (P, ρ).*

Now we manufacture a sheaf out of the collections of well-formed processes. For the sake of simplicity we do not follow [10], where such a functor is obtained as the carrier of the initial algebra for a signature endofunctor, but we give an explicit definition. Our syntactic endofunctor $N : \mathbf{G} \to \mathbf{Set}$ is given by

$$Ng := \{p \text{ well-formed} \mid \mathrm{fn}(p) \subseteq \mathcal{N}g\}/\equiv \qquad N(\sigma : g \to g') := \lambda p \in Ng.p\widetilde{\sigma}$$

where $\widetilde{\sigma}$ is the extension of $\mathcal{N}\sigma$ to processes. For the purpose of defining the NCPi $\mathbf{G}_\mathbf{I}$-IL$_{nc}$TS, we just need the functor

$$N_\mathbf{I} := \mathscr{R}N \quad \text{where} \quad \mathscr{R} := (-) \circ (\mathbf{G}_\mathbf{I} \hookrightarrow \mathbf{G}) : \mathbf{Set}^\mathbf{G} \to \mathbf{Set}^{\mathbf{G}_\mathbf{I}},$$

which only applies injective renamings. This functor is indeed a sheaf, as stated by the following theorem.

Theorem 4.10. *$N_\mathbf{I}$ preserves pullbacks.*

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

$$\frac{p \xrightarrow{ar;\bullet} p'}{g \vdash p \xrightarrow{ar;\bullet}_{\nu} g \vdash p'} \; r \in \mathcal{N}g \qquad \frac{p \xrightarrow{ab;\bullet} p'}{g \vdash p \xrightarrow{a\star;\bullet}_{\nu} \delta^{\bullet} g \vdash p'[\star/b, old^{\bullet}_{g}]} \; b \notin \mathcal{N}g$$

$$\frac{p \xrightarrow{al_{bc};\bullet} p'}{g \vdash p \xrightarrow{a\star_{bc};\bullet}_{\nu} \delta^{\bullet\to\bullet} g \vdash p'[\star_{bc}/l_{bc}, old^{\bullet\to\bullet}_{g}]} \; l_{bc} \notin \mathcal{N}g$$

$$\frac{p \xrightarrow{\bullet;W;\bar{a}(b)} p'}{g \vdash p \xrightarrow{\bullet;W;\bar{a}(\star)}_{\nu} \delta^{\bullet} g \vdash p'[\star/b, old^{\bullet}_{g}]} \qquad \frac{p \xrightarrow{\bullet;W;\bar{a}(l_{bc})} p'}{g \vdash p \xrightarrow{\bullet;W;\bar{a}(\star_{bc})}_{\nu} \delta^{\bullet\to\bullet} g \vdash p'[\star_{bc}/l_{bc}, old^{\bullet\to\bullet}_{g}]}$$

$$\frac{p \xrightarrow{\bullet;W;\bar{a}r} p'}{g \vdash p \xrightarrow{\bullet;W;\bar{a}r}_{\nu} g \vdash p'} \qquad \frac{p \xrightarrow{a;W;b} p'}{g \vdash p \xrightarrow{a;W;b}_{\nu} g \vdash p'} \qquad \frac{p \xrightarrow{\bullet;W;\bullet} p'}{g \vdash p \xrightarrow{\bullet;W;\bullet}_{\nu} g \vdash p'}$$

Fig. 6. Rules generating the transitions of $p \in N_{\mathbf{I}}g$ in the NCPi $\mathbf{G}_{\mathbf{I}}$-IL$_{nc}$TS.

The transition relation $\longrightarrow_{\nu}$ for our $\mathbf{G}_{\mathbf{I}}$-IL$_{nc}$TS is the smallest one generated by the rules in Fig. 6, which associate indices to ordinary NCPi transitions. Actually, since there are infinitely many $g \in |\mathbf{G}_{\mathbf{I}}|$ such that $p \in N_{\mathbf{I}}g$, each untyped transition has many typed counterparts. Notice the first five rules, inferring input and output paths: they collapse transitions that differ only for the fresh sent/received name to a single one exhibiting the generic fresh name $\star$ (or $\star_{bc}$). This is because the inferred transition must match the behavioral functor definition. Consequently, in the continuation the fresh name is replaced by $\star$ or $\star_{bc}$, and the names that were free in p are replaced using the colimit maps involved in the definition of $\delta^{\bullet}$ and $\delta^{\bullet\to\bullet}$, so that the resulting process has the correct index. This ensures that (i) of Definition 4.6 is satisfied. As for (ii), this comes directly from the transition system being closed under injective renamings (Proposition 3.10).

Example 4.11. Consider the transition

$$(x_{ab})(l_{ac}.\bar{a}c.\mathbf{0} \mid \bar{a}x_{ab}.x_{ab}.\mathbf{0}) \xrightarrow{\bullet;l_{ac};\bar{c}(x_{ab})} \bar{a}c.\mathbf{0} \mid x_{ab}.\mathbf{0};$$

The source process can be indexed by g of Example 4.3. Now, according to Fig. 6, the continuation should be indexed by $\delta^{\bullet\to\bullet}g$, but we have to rename it via $[a''/a, b''/b, c''/c, l''_{a''c''}/l_{ac}, \star_{a''b''}/x_{ab}]$. The resulting transition in the NCPi $\mathbf{G}_{\mathbf{I}}$-IL$_{nc}$TS is

$$g \vdash (x_{ab})(l_{ac}.\bar{a}c.\mathbf{0} \mid \bar{a}x_{ab}.x_{ab}.\mathbf{0}) \xrightarrow{\bullet;l_{ac};\bar{c}(\star_{ab})} \delta^{\bullet\to\bullet} g \vdash \overline{a''}c''.\mathbf{0} \mid \star_{a''b''}.\mathbf{0}.$$

Finally, we have the following result, which relates B-bisimulations on the NCPi B-coalgebra and a class of network conscious bisimulations.

Theorem 4.12. *$\mathbf{G}_{\mathbf{I}}$-indexed bisimulations on $(N_{\mathbf{I}}, \longrightarrow_{\nu})$ are in bijection with:*

(i) *B-bisimulations on the corresponding B-coalgebra;*
(ii) *network conscious bisimulations closed under injective renamings.*

In particular, we have that the greatest $\mathbf{G}_{\mathbf{I}}$-indexed bisimulation, the B-bisimilarity and $\sim^{NC}$ are all equivalent, thanks to Proposition 3.12.

4.4. History dependent automata

Now we prove the existence of an efficient operational model for our calculus, in the form of a HD-automaton. In order to do this, we exploit the result of [12], which tells that some classes of presheaves are equivalent to *coproducts of symmetrized representables*, i.e. representables quotiented by composition with groups of automorphisms. They generalize named sets, which are the basic building blocks of HD-automata.

Theorem 4.13. *Let $\mathbf{C}$ be a category that is small, has pullbacks, and such that all its morphisms are monic and those in $C[c, c]$ are isomorphisms, for every $c \in |C|$. Then every pullback-preserving $P \in |\mathbf{Set}^{C}|$ is isomorphic to a coproduct of symmetrized representables.*

In our case $\mathbf{C} = \mathbf{G}_{\mathbf{I}}$, which satisfies the hypothesis of Theorem 4.13. In fact, by Proposition 4.2, $\mathbf{G}_{\mathbf{I}}$ is small and has pullbacks, since pullbacks are preserved by monos; moreover, it has only monos, by definition, and it is easy to see that $\mathbf{G}_{\mathbf{I}}[g, g]$ has only isomorphisms, for each $g \in |\mathbf{G}_{\mathbf{I}}|$. Our presheaf of processes $N_{\mathbf{I}}$ preserves pullbacks, so we have a representation for it as a coproduct of symmetrized representables.

Notice that $\mathbf{G}$ does not satisfy Theorem 4.13, due to the presence of non-monic morphisms. Therefore we cannot apply the theory of [12] to the extended coalgebra we will give in Section 4.5.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

4.5. Saturated semantics

Right Kan extensions provide a canonical way of translating the NCPi transition systems from $\mathbf{Set}^{\mathbf{G_I}}$ to $\mathbf{Set}^{\mathbf{G}}$, as shown for other presheaf categories in [10,26]. The fundamental construction is the following adjunction

$$\mathbf{Set}^{\mathbf{G}} \underset{\mathscr{E}}{\overset{\mathscr{R}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathbf{Set}^{\mathbf{G_I}} \tag{1}$$

which exists because $\mathbf{G_I}$ is small and **Set** has all limits (Theorem 2.3). According to Section 2.1, $\mathscr{E} : \mathbf{Set}^{\mathbf{G_I}} \to \mathbf{Set}^{\mathbf{G}}$ can be computed pointwise as a limit in **Set**:

$$\mathscr{E}P(g) = \left\{ t \in \prod_{\sigma : g \to g' \in \|\mathbf{G}\|} P(g') \;\middle|\; \begin{array}{l} \forall \sigma_1 : g \to g_1, \\ \sigma_2 : g_1 \rightarrowtail g_2 \in \|\mathbf{G}\| : \\ \quad t_{\sigma_1}[\sigma_2] = t_{\sigma_2 \circ \sigma_1} \end{array} \right\}$$

$$\mathscr{E}P(\sigma) = \lambda t \in \mathscr{R}P(g).\{(t_{\sigma \circ \sigma'})_{\sigma'}\}_{\sigma' : g' \to g''} \quad (\sigma : g \to g')$$

In words, $\mathscr{E}P(g)$ is a set of tuples with one component for each morphism from g in $\mathbf{G}$. The tuples' components are taken from P according to the corresponding morphism's codomain, and must satisfy a "closure under monos" condition, namely: selecting the σ_1-component of a tuple and applying $P\sigma_2$ to it, where σ_2 is any applicable morphism (i.e. σ_1 and σ_2 must be composable), must yield the same result as selecting the $(\sigma_2 \circ \sigma_1)$-component of the same tuple. As for $\mathscr{E}P(\sigma)$, it takes a tuple and builds another tuple out of it, whose σ'-component is the $(\sigma \circ \sigma')$-component of the original tuple.

Unit and counit of (1) are natural transformations $\eta : Id_{\mathbf{Set}^{\mathbf{G}}} \to \mathscr{E}\mathscr{R}$ and $\varepsilon : \mathscr{R}\mathscr{E} \to Id_{\mathbf{Set}^{\mathbf{G_I}}}$. For $P : \mathbf{G} \to \mathbf{Set}$, $Q : \mathbf{G_I} \to \mathbf{Set}$ and $g \in |\mathbf{G}|$, they are given by

$$(\eta_P)_g = \lambda p \in Pg.\{(p[\sigma])_\sigma\}_{\sigma : g \to g'} \qquad (\varepsilon_Q)_g = \lambda t \in \mathscr{R}\mathscr{E}Q(g).t_{id_g}$$

The intuition, in terms of processes, is that $(\eta_P)_g$ maps a process $p \in Pg$ to a tuple obtained by applying every possible renaming $\sigma : g \to g'$ to p. Viceversa, $(\varepsilon_Q)_g$ takes a tuple of processes in $\mathscr{R}\mathscr{E}Q(g)$ and extracts the one with identity index. The well-known equations relating unit and counit ensure that the operations they perform are consistent with each other: producing the tuple of all possible renamings of p and picking the identity component just yields p.

This adjunction can be exploited to define an extended behavioral functor.

Theorem 4.14. *The functor $\widehat{B} := \mathscr{E}B\mathscr{R}$ is accessible and preserves weak pullbacks.*

Moreover, (1) yields the following correspondence.

Proposition 4.15. *For every $P : \mathbf{G} \to \mathbf{Set}$, there is a bijection between $\widehat{B}$-coalgebras having P as carrier and B-coalgebras having $\mathscr{R}P$ as carrier.*

The idea is the following. Given a B-coalgebra $(\mathscr{R}P, \rho)$, the structure map of the corresponding $\widehat{B}$-coalgebra, when applied to $p \in Pg$, builds a tuple whose σ-component is the set of transitions of $p[\sigma]$ according to ρ. Viceversa, given a $\widehat{B}$-coalgebra (P, ϕ), one can recover a B-coalgebra whose structure map gives, for each $p \in \mathscr{R}P(g)$, only the id_g-component of the tuple $\phi(p)$.

$\widehat{B}$-coalgebras can be characterized as indexed transition systems with richer labels than those of $\mathbf{G_I}$-IL$_{nc}$TSs, similarly to what was done in [26]. We call such transition systems *saturated*; this term is borrowed from [25].

Definition 4.16 *(Saturated $\mathbf{G_I}$-IL$_{nc}$TS).* Given the following presheaf of labels

$$Lab_{SAT} := \sum_{g' \in \mathbf{G}} \mathrm{Hom}_{\mathbf{G}}(-, g') \times Lab(g')$$

a *saturated* $\mathbf{G_I}$-*IL$_{nc}$TS* ($\mathbf{G_I}$-IL$_{nc}$TS$_{SAT}$) is a $\mathbf{G}$-ILTS $(P, \longrightarrow)$ with labels in $\int Lab_{SAT}$, such that:

(i) Transitions are of the form $g \vdash p \xrightarrow{(\sigma,\alpha)} g'' \vdash p'$, where $\sigma \in \mathbf{G}[g, g']$,$\alpha \in Lab(g')$ and $g'' \vdash p'$ is a valid continuation for α according to (ii) of Definition 4.6;

(ii) Transitions are such that, for all $\sigma : g \to g'$ in $\mathbf{G}$:

(a) for all $\sigma' : g' \rightarrowtail g''$ and $\varphi \in \{Id, \delta^{\bullet}, \delta^{\bullet \to \bullet}\}$, $g \vdash p \xrightarrow{(\sigma,\alpha)} \varphi(g') \vdash p'$ if and only if $g \vdash p \xrightarrow{(\sigma' \circ \sigma, \alpha[\sigma']_{Lab})} \varphi(g'') \vdash p'[\varphi(\sigma')]$ (closure under monos);

(b) $g \vdash p \xrightarrow{(\sigma' \circ \sigma, \alpha)} g'' \vdash p'$ if and only if $g' \vdash p[\sigma] \xrightarrow{(\sigma', \alpha)} g'' \vdash p'$ (transitions are preserved and reflected by morphisms);

$$
\begin{array}{rl}
p & ::= \mathbf{0} \mid \pi.p \mid p + p \mid p \mid p \mid (r)p \mid A(r_1, r_2, \ldots, r_n) \\
\pi & ::= \bar{a}br \mid a(s) \mid l_{ab} \mid \tau \\
r & ::= a \mid l_{ab} \\
s & ::= a \mid l_{(ab)} \\
\multicolumn{2}{l}{A(s_1, s_2, \ldots, s_n) \stackrel{\text{def}}{=} p \qquad i \neq j \Longrightarrow \mathrm{n}(s_i) \cap \mathrm{n}(s_j) = \varnothing}
\end{array}
$$

Fig. 7. Syntax of κNCPi processes.

Proposition 4.17. *$\widehat{B}$-coalgebras are in bijection with $\mathbf{G}_\mathbf{I}$-$IL_{nc}TS_{SAT}$s.*

Behavioral equivalences for $\mathbf{G}_\mathbf{I}$-$\mathrm{IL}_{nc}\mathrm{TS}_{SAT}$s, according to Definition 2.10, now are indexed over $\mathbf{G}$: condition (ii), for every such a relation R, requires that, whenever $(p, q) \in R_g$, then $(p[\sigma], q[\sigma]) \in R_{g'}$, for all $\sigma : g \to g'$ in $\mathbf{G}$. In other words, we have closure under *all* renamings.

Now, consider the NCPi $\mathbf{G}_\mathbf{I}$-IL_{nc}TS. Proposition 4.15 can be equivalently restated on indexed transitions systems: it allows us to derive a $\mathbf{G}_\mathbf{I}$-$\mathrm{IL}_{nc}\mathrm{TS}_{SAT}$ $(N, \longrightarrow_{\nu\text{-}SAT})$ as follows

$$\frac{\sigma : g \to g' \qquad p \in Pg \qquad g' \vdash p[\sigma] \xrightarrow{\alpha}_\nu g'' \vdash p'}{g \vdash p \xrightarrow{(\sigma,\alpha)}_{\nu\text{-}SAT} g'' \vdash p'}$$

Now it is clear why we use the term "saturated": $\mathbf{G}_\mathbf{I}$-$\mathrm{IL}_{nc}\mathrm{TS}_{SAT}$s are the saturated (according to [29]), but equivalent, version of the corresponding $\mathbf{G}_\mathbf{I}$-IL_{nc}TSs, with contexts being the morphisms of $\mathbf{G}$. To the best of our knowledge, the fact that right-Kan-extending amounts to saturating has been first observed in [26].

As discussed in Section 4.3, in general $\mathbf{G}$-indexed bisimulations and $\widehat{B}$-bisimulations do not fully correspond, unless the presheaf of states is indeed a sheaf. As stated in [24, Proposition 4.1.3], it is enough to check that $\mathscr{R}N$, i.e. $N_\mathbf{I}$, is a sheaf. This is indeed the case (Theorem 4.10), so we have the following.

Theorem 4.18. $\mathbf{G}$*-indexed bisimulations on $(N, \longrightarrow_{\nu\text{-}SAT})$ are in bijection with:*

(i) *$\widehat{B}$-bisimulations on the corresponding $\widehat{B}$-coalgebra;*
(ii) *network conscious bisimulations closed under all renamings.*

In particular, we have that the $\widehat{B}$-bisimilarity characterizes the greatest network conscious bisimulation that is closed under all renamings. This, by Theorem 3.15, is a congruence w.r.t. input prefix, but not parallel composition. See note added in proof for a discussion.

5. Concurrent NCPi

In this section we present an "enhanced" version of NCPi, called *concurrent NCPi* (κNCPi), where

- the input primitive is more flexible: it can express the reception of a link together with its endpoints;
- the output primitive is closer to actual routing protocols: it also specifies the *destination site*;
- observations represent concurrent transmissions in the form of *multisets* of routing paths.

The main result is that bisimilarity for the new semantics is a congruence. This is due to the richer observations that make the bisimilarity finer and compositional.

5.1. Syntax

The syntax of κNCPi processes is given in Fig. 7. For convenience, we distinguish names that can be output or restricted (syntactic category r) and those that can be input or can be formal parameters of process definitions (syntactic category s); $l_{(ab)}$, belonging to the latter category, denotes a link whose endpoints are both bound and we let $\mathrm{n}(l_{(ab)}) := \{l_{ab}, a, b\}$.

Input and output prefixes have the following forms: $\bar{a}br$ means that $\bar{a}br.p$ can emit the datum r, having destination b, at a and continue as p; $a(s)$ means that $a(s).p$ can receive at a a datum to be bound to s and continue as p. The intended meaning of $c(l_{(ab)}).p$ is an atomic, polyadic version of $c(a).c(b).c(l_{ab}).p$.

The definition of $\mathrm{fn}(p)$ for the new constructs is

$$\mathrm{fn}(\bar{a}br.p) := \{a, b\} \cup \mathrm{n}(r) \cup \mathrm{fn}(p)$$
$$\mathrm{fn}(a(l_{(bc)}).p) := \{a\} \cup \mathrm{fn}(p) \setminus (\{b, c\} \cup \mathcal{L}_b \cup \mathcal{L}_c)$$

Now we introduce the notion of *well-formedness* for κNCPi processes. The only additional condition w.r.t. Definition 3.2 concerns the input prefix.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

α-equivalence:

$$(a)p \equiv (a')p[a'/a] \qquad b(a).p \equiv b(a').p[a'/a] \qquad a' \# (a)p$$

$$\begin{array}{ll} (l_{ab})p \equiv (l'_{ab})p[l'_{ab}/l_{ab}] & l'_{ab} \# (l_{ab})p \\ a(l_{(bc)}).p \equiv a(l'_{(b'c')}).p[l'_{b'c'}/l_{(bc)}] & b', c', l'_{b'c'} \# a(l_{(bc)}).p \end{array}$$

Unfolding law:

$$A(r_1, \ldots, r_n) \equiv p[r_1/s_1, \ldots, r_n/s_n] \quad \text{if } A(s_1, \ldots, s_n) \stackrel{\text{def}}{=} p$$

Fig. 8. Structural congruence for well-formed κNCPi processes.

Paths

$$\begin{array}{rcl} \alpha & ::= & a; W; b \mid \bullet; W; \bullet \mid \bullet; W; \bar{a}br \mid abr; W; \bullet \\ & & \mid ab(s); W; \bullet \qquad \mathrm{n}(s) \cap \big(\mathrm{n}(W) \cup \{a, b\}\big) = \varnothing \\ W & ::= & l_{ab} \mid W; W \mid \varepsilon \\ r & ::= & a \mid l_{ab} \qquad s ::= a \mid l_{(ab)} \end{array}$$

Concurrent paths

$$\Lambda ::= \mathbf{1} \mid \alpha \mid \Lambda_1 | \Lambda_2 \mid (r)\Lambda$$

Fig. 9. Syntax of concurrent paths.

Definition 5.1 *(Well-formed κNCPi process).* A κNCPi process p is *well-formed* if every subterm q satisfies requirements (i) and (ii) of Definition 3.2 and, moreover, (iii) $q = c(l_{(ab)}).p'$ implies $\mathrm{fn}(q) = \{c\} \cup \mathrm{fn}(p') \setminus \{a, b, l_{ab}\}$.

Structural congruence, shown in Fig. 8, is minimal: we only have α-conversion and unfolding. Other axioms, such as the monoidality of $|$, will be replaced by a suitable structural congruence on observations or, in the case of scope extension, by an explicit scope closure rule.

5.2. Concurrent semantics

Observations for the concurrent semantics, defined in Fig. 9, are multisets of paths, called *concurrent paths*. For the purpose of describing a more realistic network behavior, we equip paths α with some additional information:

- paths always specify a destination site;
- both input and output paths exhibit a list of links; in the case of input paths, they are the links that can be potentially traversed in order to reach the destination;
- there is a *bound input path*, which represents the reception of a bound name; this is needed because the concurrent semantics has an explicit scope closure rule;
- there is no extrusion path, because extrusions will be represented via concurrent paths, as we allow many paths to extrude the same name.

Concurrent paths can have the following forms:

- the *empty concurrent path* 1 indicates that no activity is performed;
- the *singleton concurrent path* α is a concurrent path made of a single path;
- the *union* $\Lambda_1 | \Lambda_2$ means that the paths in Λ_1 and Λ_2 are being traversed *at the same time*;
- the *extrusion restriction* $(r)\Lambda$ indicates that r is being extruded through one or more paths in Λ.

We shall use W_α to denote the sequence of links of α and $|W_\alpha|$ to denote the set of links appearing in W_α. The set $\mathrm{fn}(\alpha)$ is redefined to include the destination site of α; if α is a bound input path, then $\mathrm{bn}(\alpha)$ is $\mathrm{n}(s)$, otherwise it's the empty set. We also define

$$\mathrm{obj}(\bullet; W; \bar{a}br) := \mathrm{obj}(abr; W; \bullet) := \{b\} \cup \mathrm{n}(r) \quad \mathrm{obj}(ab(s); W; \bullet) := \{b\}$$

This is analogous to actual routing, where a payload and its destination address travel together within a packet.

Given a concurrent path Λ, the functions $\mathrm{Is}(\Lambda)$, $\mathrm{Fn}(\Lambda)$, $\mathrm{Bn}(\Lambda)$, $\mathrm{Obj}(\Lambda)$ and $\mathrm{Obj}_{\mathrm{in}}(\Lambda)$ are the extensions to multisets of the corresponding functions on single paths. They are defined as expected, but we have to be careful with the following cases:

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

- Monoidal axioms for ";", with ϵ as identity, and for $\mid$ (plus commutativity), with 1 as identity;
- Scope extension axioms:

$$\begin{array}{ll} (r)(r')\Lambda \equiv_\Lambda (r')(r)\Lambda & r \notin \mathrm{n}(r') \\ \Lambda_1 \mid (r)\Lambda_2 \equiv_\Lambda (r)(\Lambda_1 \mid \Lambda_2) & r \# \Lambda_1 \end{array}$$

Fig. 10. Structural congruence $\equiv_\Lambda$ for well-formed concurrent paths.

$$\mathrm{Fn}((a)\Lambda) = \mathrm{Fn}(\Lambda) \setminus (\{a\} \cup \mathcal{L}_a) \qquad \mathrm{Bn}((a)\Lambda) = \mathrm{Bn}(\Lambda) \cup \{a\} \cup (\mathcal{L}_a \cap \mathrm{n}(\Lambda))$$

that are treated analogously for the other functions.

Observations here have a more complicated binding structure than those of NCPi, so we introduce a notion of well-formedness for them. Moreover, in order to be closer to actual routing protocol, we only admit *simple* paths, i.e. those that do not go through the same link twice.

Definition 5.2 *(Well-formed, canonical, simple concurrent paths).* Let Λ be a concurrent path. Then it is:

- *well-formed* if for every subterm Λ' of the form $(a)\Lambda''$ we have $\mathrm{Fn}(\Lambda') = \mathrm{Fn}(\Lambda'') \setminus \{a\}$;
- in *canonical form* if it has the form $(R)\Theta$, where R is a sequence of restrictions and Θ does not contain extrusion restrictions (binders of the form $ab(s)$ are still allowed in Θ);
- *simple* if, for all $\alpha \in \Lambda$, each $l_{ab} \in |W_\alpha|$ appears in W_α once.

An example of non-well-formed, non-simple concurrent path is $(d)(\bullet; l_{ab}; l'_{ba}; l_{ab}; \bar{b}cd \mid a; l''_{ad}; d)$, because (d) implicitly binds l''_{ad} and there are two occurrences of l_{ab} in the first path.

Simplicity is just one of the possible conditions. In general, one might want to express more complex requirements and apply static analysis methods to check them. This can be achieved through suitable type systems. For instance, QoS requirements could be expressed by associating quantitative information to links.

Well-formed paths are subject to some structural congruence axioms, shown in Fig. 10. They establish that paths are strings and concurrent paths are multisets, and that extrusion restrictions can be swapped and grouped at the outest level. Scope extension requires some side conditions in order to avoid captures and enforce well-formedness. Clearly, any concurrent path satisfying these conditions can be converted to canonical form.

We write $\Lambda /\!\!/ r$ for the operation that applies $/r$ to each $\alpha \in \Lambda$ if $r \notin \mathrm{bn}(\Lambda)$, yields Λ otherwise. We use the symbols $r \#_\Lambda \Lambda$, extended to sets of names as expected, to mean $r \notin \mathrm{n}(\Lambda)$.

Definition 5.3 *(κNCPi transition system).* The κNCPi transition system is the smallest transition system generated by the rules in Fig. 11, where observations are up to $\equiv_\Lambda$ and transitions are closed under $\equiv$, i.e. if $p \overset{\Lambda}{\Longrightarrow} q$, $p \equiv p'$, $q \equiv q'$ and $\Lambda \equiv_\Lambda \Lambda'$, then $p' \overset{\Lambda'}{\Longrightarrow} q'$.

Axioms (IN) and (OUT) are similar to those of Fig. 5, but they infer a path with an explicit destination; this site is the same as the reception one for (IN), because the paths it infers have length 0. We also have (BIN), which infers an input with bound placeholder. Axioms (INT) and (LINK) are the same as the interleaving case, so they are omitted. The axiom (IDLE) infers a "no-op" transition, enabling the parallel composition of processes to behave in an interleaving style.

The rule (SUM-L) is obvious. It has a right counterpart, because $+$ is not commutative for κNCPi. This rule is omitted in Fig. 11.

The rule (RES) and (OPEN) are an obvious extension of those in Fig. 5. Notice that (OPEN) allows one to "extrude" the destination site: the intuition is that we can use global resources to send or receive a datum to/from a local site, which becomes global if the communication is not complete.

The rule (PAR) makes the union of two concurrent paths, but only if bound names of each concurrent path are fresh w.r.t. the other process and do not occur in the other path. This last condition avoids inferring transitions where the extruded name is free in the receiving process's continuation even if it has not been actually received, which might cause incorrect behaviors. For instance, consider the processes

$$p \overset{\text{def}}{=} (b)\bar{a}ab.b(c).p' \qquad q \overset{\text{def}}{=} a(d).\bar{d}de.q'$$

and suppose the following transition is allowed

$$p \mid q \xrightarrow{(b)\bullet;\bar{a}ab \mid aab;\bullet} b(c).p' \mid \bar{b}be.q'[b/d].$$

Now the two components of the continuation can synchronize on b even if its scope extension has not actually been accomplished, which is clearly incorrect.

The remaining rules are used to synchronize processes. The synchronization is performed in two steps:

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

$$\text{(IN)}\ a(s).p \xrightarrow{aar;\bullet} p[r/s] \qquad \text{(BIN)}\ a(s).p \xrightarrow{aa(s);\bullet} p \qquad \text{(OUT)}\ \overline{a}br.p \xrightarrow{\bullet;\overline{a}br} p \qquad \text{(IDLE)}\ p \overset{1}{\Rightarrow} p$$

$$\text{(SUM-L)}\quad \frac{p \overset{\Lambda}{\Rightarrow} p'}{p+q \overset{\Lambda}{\Rightarrow} p'}$$

$$\text{(RES)}\quad \frac{p \overset{\Lambda}{\Rightarrow} q}{(r)p \xrightarrow{\Lambda /\!/ r} (r)q} \qquad r \notin \mathrm{Obj}(\Lambda) \cup \mathrm{Bn}(\Lambda) \cup \mathrm{Is}(\Lambda)$$

$$\text{(OPEN)}\quad \frac{p \overset{\Lambda}{\Rightarrow} q}{(r)p \xrightarrow{(r)(\Lambda /\!/ r)} q} \qquad r \in \mathrm{Obj}(\Lambda) \setminus (\mathrm{Is}(\Lambda) \cup \mathrm{Obj}_{\mathrm{in}}(\Lambda))$$

$$\text{(PAR)}\quad \frac{p_1 \overset{\Lambda_1}{\Rightarrow} q_1 \quad p_2 \overset{\Lambda_2}{\Rightarrow} q_2}{p_1 \,|\, p_2 \xrightarrow{\Lambda_1 \,|\, \Lambda_2} q_1 \,|\, q_2} \qquad \begin{array}{l} \mathrm{Bn}(\Lambda_i) \# p_{3-i} \\ \mathrm{Bn}(\Lambda_i) \#_\Lambda \Lambda_{3-i} \\ i=1,2 \end{array}$$

$$\text{(COM)}\quad \frac{p \xrightarrow{(R)\ (\bullet;W;\overline{a}br \,|\, ab'x;W';\bullet \,|\, \Theta)} q}{p \xrightarrow{(R')\ (\bullet;W;W';\bullet \,|\, \Theta)} (R'')\, q(\sigma_b \circ \sigma_r)} \qquad \begin{array}{l} R' = R \cap \mathrm{Obj}(\Theta) \\ R'' = (R \setminus R') \cap (\{b\} \cup \mathrm{n}(r)) \\ \text{see tables (b) and (c)} \end{array}$$

$$\text{(SRV-IN)}\quad \frac{p \xrightarrow{(R)\ (a;W;b \,|\, bcx;W';\bullet \,|\, \Theta)} q}{p \xrightarrow{(R)\ (acx;W;W';\bullet \,|\, \Theta)} q}$$

$$\text{(SRV-OUT)}\quad \frac{p \xrightarrow{(R)\ (\bullet;W;\overline{a}br \,|\, a;W';c \,|\, \Theta)} q}{p \xrightarrow{(R)\ (\bullet;W;W';\overline{c}br \,|\, \Theta)} q}$$

$$\text{(SRV-SRV)}\quad \frac{p \xrightarrow{a;W;b \,|\, b;W';c \,|\, \Lambda} q}{p \xrightarrow{a;W;W';c \,|\, \Lambda} q}$$

The concurrent path inferred by (COM), (SRV-IN), (SRV-OUT) and (SRV-SRV) must be simple.

(a)

• $b' = b$ • $\sigma_b = id$	• $b, b' \in R$ • $\sigma_b = [b/b']$

(b)

• $r \notin R$ • $x = r$ • $\sigma_r = id$	• $r \in R$ • $x = (s)$ • $\sigma_r = [r/s]$

(c)

Fig. 11. κNCPi SOS rules: (a) shows the SOS rules; (b) and (c) are the possible configurations for (COM). Any pair of configurations, one from (b) and one from (c), is valid (four possibilities).

(i) paths of parallel processes are collected through the rule (PAR);
(ii) (COM), (SRV-IN), (SRV-OUT) and (SRV-SRV) take two compatible paths out of the resulting multiset and replace them with their concatenation, without modifying the source process; in other words, these rules synchronize two subprocesses of the source process.

The rule (COM) covers all kinds of communications, yielding a complete path. In the case of extrusions, input placeholders in the continuation are replaced with extruded names; these are removed from the resulting transition's label, provided that there are no other paths extruding them. For instance, consider the process

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

$$p \stackrel{\text{def}}{=} (c)(\bar{a}bc.p_1 \mid \overline{a'}b'c.p_2 \mid l_{ab}.p_3) \mid b(x).p_4;$$

its behavior can be the extrusion of c

$$p \xrightarrow{\bullet;l_{ab};\bullet} (c)(p_1 \mid \overline{a'}b'c.p_2 \mid p_3 \mid p_4[c/x])$$

but also a transition where the scope of c remains open

$$p \xrightarrow{(c)(\bullet;l_{ab};\bullet \mid \bullet;\overline{a'}b'c)} p_1 \mid p_2 \mid p_3 \mid p_4[c/x]$$

due to the presence of $\bullet;\overline{a'}b'c$ in the label.

The rules (SRV-IN), (SRV-OUT) and (SRV-SRV) allow extending a path with a service path.

The premises of (COM), (SRV-IN) and (SRV-OUT) must have their concurrent paths in canonical form: this is always possible, thanks to (PAR) side conditions.

The following proposition states that the transition system generated by these rules is well-behaved.

Proposition 5.4. *If $p \stackrel{\Lambda}{\Longrightarrow} q$ then Λ is simple and well-formed, and q is well-formed.*

The behavioral equivalence for κNCPi processes is called *concurrent network conscious bisimilarity*, and is an obvious extension of Definition 3.7.

Definition 5.5 *(Concurrent network conscious bisimulation).* A binary, symmetric and reflexive relation $\mathcal{R}$ is a *concurrent network conscious bisimulation* if $(p,q) \in \mathcal{R}$ and $p \stackrel{\Lambda}{\Longrightarrow} p'$, with $\mathrm{Bn}(\Lambda) \# q$, implies that there is q' such that $q \stackrel{\Lambda}{\Longrightarrow} q'$ and $(p',q') \in \mathcal{R}$. The bisimilarity is the largest such relation and is denoted by $\sim_\kappa^{NC}$.

Theorem 5.6. *$\sim_\kappa^{NC}$ is a congruence with respect to all κNCPi operators.*

Proof sketch. We have to prove that $\sim_\kappa^{NC}$ is closed under each operator. The difficult case is the input prefix, since a renaming, possibly not injective, is involved. The idea behind the proof is that, even though a renaming σ may enable some (COM), (SRV-IN), (SRV-OUT) or (SRV-SRV) rules in the proof of a transition of $p\sigma$, the collection of paths they concatenate is the renamed version of an observation of p, and thus of every q bisimilar to p. This is an overview of the proof steps:

(i) First, we prove that, given any transition $p \stackrel{\Lambda}{\Longrightarrow} q$ and renaming σ such that $\Lambda\sigma$ is simple, we have $p\sigma \stackrel{\Lambda\sigma}{\Longrightarrow} q\sigma$.

(ii) Consider a transition $p \stackrel{\Lambda}{\Longrightarrow} q$ and let Π be its proof. We prove that we can always bring towards the root the rule instances in Π that, in their consequences, concatenate paths α_1 and α_2 whose common interaction sites are free in p. This is done by permuting such rules with their parents in the proof tree: whenever the parent rule regard binding operators, the requirements about interaction sites ensure that these names are not involved in the rule's side conditions, so swapping the order of restriction and concatenation is allowed. The resulting proof Π' infers a transition $p' \stackrel{\Lambda}{\Longrightarrow} q'$, where $p' \equiv p$ and q' is q with some unguarded restrictions at the outest level (this may happen when an application of (COM), used in Π to close the scope of some restrictions, is delayed in Π').

(iii) Next we prove that, for any σ and p, if $p\sigma \stackrel{\Lambda}{\Longrightarrow} q$ has proof Π, and σ does not map any name to interaction sites or objects of paths concatenated throughout Π, then we can recover a transition $p \stackrel{\Lambda'}{\Longrightarrow} q'$ such that $\Lambda'\sigma = \Lambda$ and $q'\sigma = q$. In other words, if σ did not enable any rule in Π, then Π infers a transition that is the renamed version of one of p.

(iv) Given a transition $p\sigma \stackrel{\Lambda}{\Longrightarrow} q$, we can always recover a transition $p \stackrel{\Lambda'}{\Longrightarrow} q'$ from which, after applying some rules that concatenate paths in Λ', we get $p\sigma \stackrel{\Lambda}{\Longrightarrow} q^*$, where q^* is q with some unguarded restrictions at the outest level. The idea is to bring all the rule instances enabled by σ towards the root via (ii); the remaining ones satisfy the requirements of (iii), hence we can find the required transition of p.

(v) Finally, we prove that we can construct a bisimulation $\mathcal{R}$ containing all the pairs $(p\sigma, q\sigma)$, with $p \sim_\kappa^{NC} q$, which is also closed under restrictions (i.e. $((r)p, (r)q) \in \mathcal{R}$ whenever $(p,q) \in \mathcal{R}$) and contains scope extension (i.e. if q is p with some restrictions brought at the top level, then $(p,q) \in \mathcal{R}$). In fact, given $p\sigma \stackrel{\Lambda}{\Longrightarrow} p'$, we use (iv) to get a transition $p \stackrel{\Lambda'}{\Longrightarrow} p''$ and a sequence Ω of rule instances that concatenate pairs of paths. Such transition can be simulated by $q \stackrel{\Lambda'}{\Longrightarrow} q'$. Using (i) we get $q\sigma \stackrel{\Lambda'\sigma}{\Longrightarrow} q'\sigma$ and, applying a suitably adapted version of Ω, we get $q\sigma \stackrel{\Lambda}{\Longrightarrow} (R)(q'\sigma)\sigma'$, where R and σ' are added by instances of (COM) treating extrusions. Since Ω has the same effect on $p''\sigma$, i.e. it binds R and applies σ', we have that $(R)(p''\sigma)\sigma'$ is p' with some restrictions at the topmost level. Now, since $p'' \sim_\kappa^{NC} q'$, by definition of $\mathcal{R}$, and by closure under scope extension and restrictions, we can conclude $(p', (R)(q'\sigma)\sigma') \in \mathcal{R}$. □

This result allows us to equip the π-calculus with a concurrent semantics. In fact, we can characterize π-calculus processes via a syntactic restriction, as done in Definition 3.8.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

Definition 5.7 *(Concurrent linkless NCPi).* We call *concurrent linkless NCPi* (κNCPi$_{\not\ell}$) the subcalculus of κNCPi where:

- no links appear in processes;
- every occurrence of the output prefix is of the form $\bar{a}ar$.

In this calculus the usual synchronization mechanism is emulated by two steps of derivation, for instance

$$(\text{COM})\ \frac{p \xrightarrow{\bar{a}b} p' \qquad q \xrightarrow{ab} q'}{p \mid p' \xrightarrow{\tau} q \mid q'} \quad \longmapsto \quad (\text{COM})\ \frac{(\text{PAR})\ \dfrac{p \xrightarrow{\bullet;\bar{a}ab} p' \qquad q \xrightarrow{aab;\bullet} q'}{p \mid q \xrightarrow{\bullet;\bar{a}ab \mid aab;\bullet} p' \mid q'}}{p \mid p' \xrightarrow{\bullet;\bullet} q \mid q'}$$

but we also have additional concurrent observations, which lead to the following result.

Corollary 5.8 *(of Theorem 5.6). The bisimilarity on the concurrent π-calculus transition system is a congruence.*

Another evidence of this result is the classical counterexample not applying: we have $\bar{a}ar \mid a(x) \not\sim_{\kappa}^{NC} \bar{a}ar.a(x) + a(x).\bar{a}ar$, because

$$\bar{a}ar \mid a(x) \xrightarrow{\bullet;\bar{a}ar \mid aar;\bullet} \mathbf{0} \qquad \bar{a}ar.a(x) + a(x).\bar{a}ar \not\xrightarrow{\bullet;\bar{a}ar \mid aar;\bullet}$$

This result is analogous to that in [16] but, as already mentioned, there the synchronization mechanism is not faithful to the π-calculus: in [16] the synchronization channel is observed unless restricted, for instance $\bar{a} \mid a \xrightarrow{\tau_a} \mathbf{0}$, while for our calculus $\bar{a} \mid a \xrightarrow{\bullet;\bullet} \mathbf{0}$, which corresponds to τ.

6. Case study: a routing protocol

Here we give a non-trivial example of how κNCPi can be used to model a routing protocol, similar to BGP [30]. This protocol assumes that the network is composed of disjoint groups of networks, each referring to a single administrative authority, called *Autonomous Systems* (AS). Some of the ASs' routers act as *gateways* between the AS they belong to and other networks. The protocol takes care of the routing mechanism between ASs in a distributed manner: each gateway has a *routing table*, filled by the protocol, whose entries specify which is the next hop along the "best" path towards some destination; this information will be used to forward the incoming data.

In our model, both routers and hosts are represented as sites, and network connections are represented as links. The whole network is modeled as the parallel composition of some autonomous systems plus the connections among them (parameters of a recursive definition are omitted when obvious)

$$Net \stackrel{\text{def}}{=} AS_1 \mid \ldots \mid AS_k \mid Overlay \qquad Overlay \stackrel{\text{def}}{=} \ldots \mid L(l^i_{g_i h_i}) \mid \ldots$$
$$L(l_{(xy)}) \stackrel{\text{def}}{=} l_{xy}.L(l_{xy})$$

Here $L(l^i_{gh})$ is a process that recursively offers a transportation service l^i_{gh} from gateway g to h. We denote by G the set $\text{fn}(Overlay) \cap \mathcal{S}$, which contains the gateways. Notice that gateways are sites, not processes, in the style of the π-calculus; alternatively, they may be modeled as dedicated processes.

An autonomous system AS_k is

$$AS_k \stackrel{\text{def}}{=} (\mathscr{L}_k)L_k \mid A_k \qquad L_k \stackrel{\text{def}}{=} \ldots \mid L(l^{i,k}_{ab}) \mid \ldots$$

where $\mathscr{L}_k = \text{fn}(L_k) \cap \mathcal{L}$ are the *local links of* AS_k, invisible to any other AS. We have two components: L_k, which keeps providing the local services, and A_k, which is the parallel composition of generic processes using some sites of AS_k to send and receive data. We call these sites *local sites of* A_k, denoted by $Loc(A_k)$: formally they are $Loc(a(s).p) = Loc(\bar{a}br.p) = \{a\}$, the other cases are obvious. The set $Loc(AS_k)$ of local sites of AS_k is $(\text{fn}(L_k) \cap \mathcal{S}) \cup Loc(A_k)$. For these we require $Loc(AS_i) \cap Loc(AS_j) = \varnothing$, for all $i \neq j$. We write G_k for the set $Loc(AS_k) \cap G$, i.e. the set of AS_k's gateways.

Now we want to model the routing mechanism. The routing tables are modeled as a collection of functions RT_g, one for each gateway g, such that $RT_g(x)$ is a link from g to some other gateway h, representing the next hop of the best path towards x. The forwarding is implemented at the SOS level by the following additional rule for gateways[1]

$$\frac{p \xrightarrow{(R)(\bullet;W;\bar{a}xr \mid a;l_{ab};b \mid \Theta)} p'}{p \xrightarrow{(R)(\bullet;W;l_{ab};\bar{b}xr \mid \Theta)} p'}\ \varphi(a,b,x,l_{ab})$$

[1] Roles played by sites, such as "gateway", are stated informally here, but they could be formalized through a type system.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

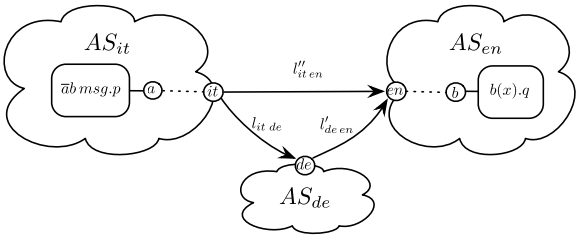


Fig. 12. Example network.

where

$$\varphi(a, b, x, l_{ab}) := a \in G_k \wedge$$
$$\textbf{if } x \in Loc(AS_k) \textbf{ then } b \in Loc(AS_k) \textbf{ else } l_{ab} = RT_a(x)$$

which means that, whenever a is a gateway of AS_k, we have two cases: if x is in AS_k then a local link must be used to extend the path, otherwise the link of the overlay network specified in a's routing table. If a is not a gateway, by definition only local links are available to extend the path. In this case the ordinary SOS rules are used.

Adding a rule to implement forwarding is more convenient than turning the routing tables into processes, which would complicate the model. In fact, one could have, for each gateway, one site for each reachable destination, and a link between two gateway sites only if they correspond to the same destination and belong to gateways involved in the optimal path toward that destination. This would rule out non-optimal complete paths.

Now, consider the network depicted in Fig. 12. We have three ASs: an Italian one, a German one and an English one; and two processes willing to communicate from AS_{it} to AS_{en}. Suppose the routing tables are such that $RT_{it}(b) = l_{it\,de}$ and $RT_{de}(b) = l'_{de\,en}$. A possible transition is

$$AS_{it} \mid AS_{en} \mid AS_{de} \mid Overlay \xrightarrow{\bullet; l_{it\,de}; l'_{de\,en}; \bullet} AS'_{it} \mid AS'_{en} \mid AS_{de} \mid Overlay.$$

Notice that only the part of the path between the gateways is observable.

Remark 6.1. The concurrent nature of the semantics here does not play a relevant role. The important element is the presence of information about destinations, which NCPi observations do not have. We could imagine a version of NCPi which is still interleaving, but paths are singleton paths of κNCPi. We did not make this choice at the very start in order to be as closer as possible to the π-calculus.

7. Conclusions

In this paper we presented NCPi, an extension of π-calculus with an explicit notion of network. To achieve this, we enriched the syntax with named connectors and defined a semantics whose observations are routing paths. Then we gave operational models for our calculus, in terms of presheaf-based coalgebras: one characterizing the ordinary observational equivalence, with an equivalent HD-automaton, and a saturated one, also closed under input prefix, but not under parallel composition (see note added in proof). Finally, we introduced concurrency in our calculus by letting observations be multisets of paths. Thanks to these additional observations, the bisimilarity becomes a congruence. In [18] concurrent NCPi is used to model the peer-to-peer architecture Pastry. The advantage is that it is possible to observe a whole routing path through the overlay as the result of multiple synchronizations among peers.

One critical point was characterizing resource allocation strategies as endofunctors on $\mathbf{Set}^{\mathbf{G}}$. This can be done directly, or by going through endofunctors on $\mathbf{G}$, which then are lifted by precomposition. We chose the latter option, because it allows us to easily obtain some good properties: the existence of Kan extensions along those functors guarantees that their lifting to presheaves preserve limits and colimits, which is essential for coalgebras employing these functors. However, in the case of edge allocation, many unnecessary resources are generated, namely edges between all pairs of vertices not involved in the allocation. Actually, since one can always recover the minimal support thanks to sheaf property of N_{I}, this is not really an issue. The other option would be having an endofunctor of the form

$$\Delta^{\bullet\to\bullet} P(g) := \sum_{\ell:[2]\to g} P(g_\ell)$$

where ℓ picks a pair of vertices in g and g_ℓ is g with a new edge between those vertices. On the one hand, this avoids wasting resources, because it generates processes indexed by graphs with just one additional edge, but on the other hand it has the conceptual disadvantage of using "implementation details" of resources at the level of syntax and semantics. Moreover, such operator does not have clear properties.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

Related work The works most closely related to ours are [7] and [8] where network-aware extensions of $D\pi$ [31] and KLAIM[32] are presented, called respectively $D\pi_F$ and TKLAIM. KLAIM is quite far from the synchronous π-calculus, because it models a distributed tuple-space modifiable through asynchronous primitives, but an encoding to the asynchronous π-calculus exists [33]. Both $D\pi_F$ and TKLAIM are *located* process calculi, which means that processes are deployed in *locations*, modeling physical network nodes. In κNCPi, instead, processes access the network through sites, possibly more than one for each process, rather than being inside of it. However, locations can be easily introduced in κNCPi by a typing mechanism which limits the number of subject names in processes. The network representations are quite different: in $D\pi_F$ locations are explicitly associated with their connectivity via a type system, TKLAIM has a special process to represent connections, while in our calculus connections are just names, so the available network nodes and connections correspond to the standard notion of free names. This brings simpler primitives, but also a higher level of dinamicity: connections can be created and passed among processes, as shown in the introductory example; this example, in our opinion, is not easily implementable in TKLAIM and $D\pi_F$. Finally, our calculus is more programmable: processes explicitly activate transportation services over connections via the link prefix, while in the cited calculi the network is always available.

We can also cite [34–36] as examples of calculi where resources carry some extra information: they explicitly associate costs with π-calculus channels through a type system. In our case, links could also be typed in order to model services with different features, e.g. performance, costs and access rights.

Besides the π-calculus, other calculi have been equipped with a presheaf-based semantics: the open π-calculus in [37], where processes are indexed by structured sets of names that represent distinctions; the explicit fusion calculus in [26], where processes are indexed by fusions in the form of equivalence classes of names; and the fusion calculus in [38], where the author uses the same presheaf category as [10] and incorporates fusions in the behavioral functor.

Future directions We plan to extend NCPi by adding other pieces of information to sites and links, e.g. *access rights*, and study the corresponding presheaf semantics. The idea is that, since the category of resources can be constructed as a category of algebras, as we did for graphs, associating more information to resources means adding sorts (objects) and operations (morphisms) to the category describing the algebraic specification. For instance, we could have an operation $r_e : e \to a$ that associates access rights to edges. Sorts and operations should be interpreted in suitable domains, e.g. we may want access rights to have a lattice structure.

An idea we plan to investigate is using presheaves to formally express the relation between the architecture and the detailed design of systems. In fact, one could think of a system as a number of components, deployed in different places, running in parallel. Forming a category of structures that model possible architectures, e.g. hypergraphs where hyperedges represent parallel components as in [39], and then use it as index category of presheaves, would give a formal way of associating the structure "in the large" and the structure "in the small" of systems. Since execution steps may create or reveal components, there will be allocation operators that refine the architecture, e.g. by replacing a single hyperedge with a number of hyperedges connecting the same nodes.

Another line of research regards HD-automata: an open issue is how to construct HD-automata for observational congruence; we argue that this can be done, in the style of the saturation construction described in Section 4.5, by explicitly introducing fusions in the model, for instance as additional transitions labeled by fusions.

8. Note added in proof

Bisimilarity for interleaving NCPi is not closed under parallel composition. This is surprising, as π-calculus extensions usually retain closure under this operator. The issue is discussed in [18]. Here we summarize this discussion.

The cause of the problem is not the language itself, but its novel synchronization mechanism: it is "transactional", meaning that a single observation may be the result of multiple synchronizations. The intuition behind the problem is that, even if a parallel system is indistinguishable from its interleaving implementation, adding a router may allow the former to build longer paths. The transactional mechanism does not affect compositionality of the concurrent semantics, because parallel and interleaving processes can always be distinguished.

Example 8.1. Consider the following processes

$$p = l_{ab} \mid l'_{cd} \qquad q = l_{ab}.l'_{cd} + l'_{cd}.l_{ab},$$

the latter being the interleaving unrolling of the former. We have $p \sim^{NC} q$, but if we put l''_{bc} in parallel to both processes we have

$$p \mid l''_{bc} \xrightarrow{a;l_{ab};l'''_{bc};l''_{cd};d} \mathbf{0}, \qquad q \mid l''_{bc} \not\xrightarrow{a;l_{ab};l'''_{bc};l''_{cd};d}$$

so $p \mid l''_{bc} \not\sim^{NC} q \mid l''_{bc}$.

However, closure under renamings does help. Let $\approx_{NC}$ denote the greatest network conscious bisimulation closed under all renamings. Then $\approx_{NC}$ is able to distinguish the processes p and q of Example 8.1. In fact, if we take any renaming σ that maps b to c, l_{ab} to $\tilde{l}_{ac}$ and l'_{cd} to $\tilde{l}'_{cd}$, we have

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

$$p\sigma \xrightarrow{a:\tilde{l}_{ac};\tilde{l}'_{cd};d} \mathbf{0} \quad \text{but} \quad q\sigma \not\xrightarrow{a:\tilde{l}_{ac};\tilde{l}'_{cd};d}.$$

The discriminative power of $\approx_{NC}$ is still not enough, as the following example shows.

Example 8.2. For the following processes

$$p' = \bar{a}r \mid l_{ab} \mid c(x)$$
$$q' = \bar{a}r \mid ((l_{ab} \mid c(x)) + (l_{ab}.c(x) + c(x).l_{ab}))$$

we have $p' \approx_{NC} q'$ but again $p' \mid l'_{bc} \xrightarrow{\bullet:l_{ab};l'_{bc}:\bullet} \mathbf{0}$, which $q' \mid l'_{bc}$ cannot simulate. This is because $l_{ab} \mid c(x)$ is within a sum in q', so it is forced to interact in an interleaving manner with $\bar{a}r$. This example suggests that some combinations of parallel and sum should be forbidden.

Finding a suitable syntactic restriction for which $\approx_{NC}$ is a congruence remains an open problem. We conjecture that a good candidate is NCPi with *guarded sums*, where prefixes and sums are replaced with

$$\sum_{i=1,\ldots,n} \pi_i.p_i \quad (n \in \mathbb{N}).$$

In fact, we only have processes that are structurally congruent to:

$$(R)\left(\sum_{i=1,\ldots,n_1} \pi_i^1.p_i^1 \mid \ldots \mid \sum_{i=1,\ldots,n_k} \pi_i^k.p_i^k\right)$$

where R is a sequence of restrictions; q' in Example 8.2 is not of this form, so it is ruled out. This supports the validity of the following conjecture.

Conjecture 8.3. *$\approx_{NC}$ is a congruence for NCPi with guarded sums.*

Acknowledgements

The authors would like to thank Vincenzo Ciancia, Fabio Gadducci and Andrea Corradini for their valuable comments and suggestions.

Appendix A. Proofs

Convention A.1. *When considering a collection of renamings and processes, we assume that processes have bound names that are distinct from their free names and from the names involved in renamings.*

Proof of Proposition 3.6. By induction on the inference. The only relevant case is whenever $p \xrightarrow{\alpha} q$ is inferred through (RES). We have $p \equiv (r)p'$, $\alpha \equiv_\alpha \alpha'/r$, $q \equiv (r)q'$ and the transition is inferred from $p' \xrightarrow{\alpha'} q'$. Since $\equiv$ preserves well-formedness, $(r)p'$ is well-formed as well and so is p'. Therefore, by induction hypothesis, q' is well-formed. Now we have to show that $(r)q'$ is well-formed. Suppose it is not, then there must be l_{ab} in $\mathrm{fn}(q')$ such that $r = a$ or $r = b$. Then, since l_{ab} must also be in $\mathrm{fn}(p')$, $(r)p'$ would not be well-formed, a contradiction. □

Proof of Proposition 4.2. Let **F** the small category of finite ordinals and functions. Being a skeleton of **FinSet**, **F** is equivalent to it. Let $F : \mathbf{F} \to \mathbf{FinSet}$ be the fully faithful and essentially surjective functor for such equivalence. Then it is easy to check that the functor $F \circ (-) : \mathbf{F}^{\rightrightarrows} \to \mathbf{FinSet}^{\rightrightarrows}$ is fully faithful and essentially surjective as well. This means that $\mathbf{F}^{\rightrightarrows}$ and $\mathbf{FinSet}^{\rightrightarrows}$ are equivalent, so $\mathbf{FinSet}^{\rightrightarrows}$ is essentially small and its skeletal category **G** is small.

As for finite colimits and pullbacks: the category of finite graphs $\mathbf{FinSet}^{\rightrightarrows}$ has them (it is a *topos* [40]), so also **G** does, by equivalence. □

Proof of Theorem 4.5. $\mathcal{P}_c \circ (-)$ is known to be accessible and preserve weak pullbacks, because so does $\mathcal{P}_c$; also $\mathcal{S}$ and $\mathcal{L}$ do, because they can be seen as constant functors; sum and products of accessible and weak-pullback-preserving functors are accessible and preserve weak pullbacks; $\Delta^\bullet$ and $\Delta^{\bullet\to\bullet}$ have both left and right adjoints, namely functors computing left and right Kan extensions along $\delta^\bullet$ and $\delta^{\bullet\to\bullet}$, so they preserve limits, in particular weak pullbacks, and (filtered) colimits. □

Proof of Proposition 4.7. Given a coalgebra (P, ρ), the equivalent $\mathbf{G_I}$-IL$_{nc}$TS $(P, \longrightarrow_\rho)$ is given by

$$g \vdash p \xrightarrow{\alpha}_\rho g' \vdash p' \iff (\alpha, p' \in Pg') \in \rho_g(p)$$

In fact, (i) of Definition 4.6 reflects the definition of B, (ii) the naturality of ρ. □

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

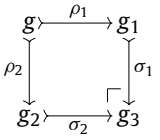
Proof of Proposition 4.8. Given a B-bisimulation (R, ρ), we shall show that $\{Rg\}_{g \in |\mathbf{G}_\mathrm{I}|}$ is a $\mathbf{G}_\mathrm{I}$-indexed bisimulation.

Consider any $g \in |\mathbf{G}_\mathrm{I}|$ and a pair $(p, q) \in Rg$. Since it is also a B-coalgebra, by Theorem 4.7 there is a $\mathbf{G}_\mathrm{I}$-IL$_{nc}$TS $(R, \longrightarrow_\rho)$ which has one transition of the form

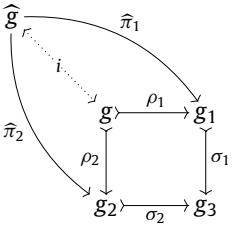
$$g \vdash (p, q) \xrightarrow{\alpha}_\rho g' \vdash (p', q'), (p', q') \in Rg',$$

for each transition of p. This gives the simulating transition of q required by (i) of Definition 2.10. Condition (ii) just amounts to saying that R is a presheaf $\mathbf{G}_\mathrm{I} \to \mathbf{Set}$. □

Proof of Theorem 4.10. Let



be any pullback in $\mathbf{G}_\mathrm{I}$. It is a pullback also in $\mathbf{FinSet}^{\rightrightarrows}$. Therefore there is a unique isomorphism i such that the following diagram in $\mathbf{FinSet}^{\rightrightarrows}$ commutes

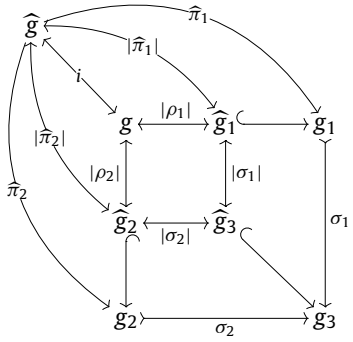


where

$$\begin{aligned}
\widehat{g} = \{ \quad & \{(e_1, e_2) \in e_{g_1} \times e_{g_2} \mid \sigma_1(e_1) = \sigma_2(e_2)\}, \\
& \{(v_1, v_2) \in v_{g_1} \times v_{g_2} \mid \sigma_1(v_1) = \sigma_2(v_2)\}, \\
& s_{\widehat{g}} := \lambda(e_1, e_2).(s_{g_1}(e_1), s_{g_2}(e_2)), \\
& t_{\widehat{g}} := \lambda(e_1, e_2).(t_{g_1}(e_1), t_{g_2}(e_2)) \quad \}
\end{aligned}$$

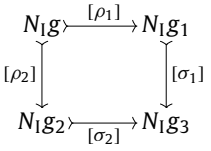
and $\widehat{\pi}_1$, $\widehat{\pi}_2$ are the restrictions to $\widehat{g}$ of the projections from $g_1 \times g_2$.

Let $\widehat{g}_1$, $\widehat{g}_2$ and $\widehat{g}_3$ denote $\widehat{\pi}_1(\widehat{g})$, $\widehat{\pi}_2(\widehat{g})$ and $\sigma_1(\widehat{g}_1)$ $(= \sigma_2(\widehat{g}_2))$, respectively. Then we can decompose each morphism as an isomorphism followed by an embedding as follows



(2)

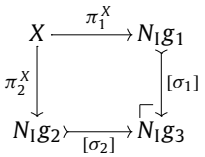
Our goal is showing that the following diagram is a pullback in **Set**



where it is easy to see that all the legs are indeed injections. In fact, they are the homomorphic extension to processes of $\mathcal{N}\rho_i$ and $\mathcal{N}\sigma_i$, for $(i = 1, 2)$, which are injective because covariant hom functors preserve monomorphisms.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

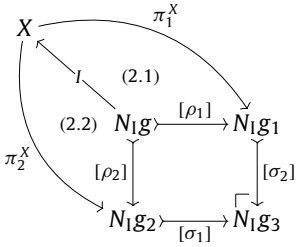
Consider the following pullback



where

$$X = \{(p_1, p_2) \in N_1g_1 \times N_1g_2 \mid p_1[\sigma_1] = p_2[\sigma_2]\}$$

and π_1^X, π_2^X are the restrictions to X of the projections from $N_1g_1 \times N_1g_2$. We shall show that the mediating morphism $I : N_1g \to X$ for the following diagram

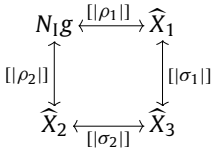


is bijective.

The idea is "lifting" the decomposition of diagram (2) to **Set**. First of all, let $\mathcal{N}_F : \mathbf{FinSet}^{\rightrightarrows} \to \mathbf{Set}$ be the functor that act as $\mathcal{N}$ on the whole $\mathbf{FinSet}^{\rightrightarrows}$ and let

$$\widehat{X}_i := \{p \in N_1g_i \mid \mathrm{fn}(p) \subseteq \mathcal{N}_F\widehat{g}_i\} \qquad i = 1, \ldots, 3$$
$$[|\rho_i|] := [\rho_i] : N_1g \to \widehat{X}_i \qquad [|\sigma_i|] := \lambda p \in \widehat{X}_i.p[\sigma_i] : \widehat{X}_i \to \widehat{X}_3 \qquad i = 1, 2$$

The functions $[|\rho_i|]$ and $[|\sigma_i|]$ are the homomorphic extension to processes of $\mathcal{N}_F|\rho_i|$ and $\mathcal{N}_F|\sigma_i|$, which are bijections, so are themselves bijective. Their definition ensure the commutativity of



(3)

Now we aim to prove that all and only the processes in $\widehat{X}_1$ and $\widehat{X}_2$ appear in X, and each process appears in only one pair. This will allows us to turn the projections from X into bijective functions by restricting their codomains to $\widehat{X}_1$ and $\widehat{X}_2$.

Formally, we have to show that:

(i) every $(p_1, p_2) \in X$ is such that $p_1 \in \widehat{X}_1$ and $p_2 \in \widehat{X}_2$;
(ii) for all $p_1 \in \widehat{X}_1$ (resp. $p_2 \in \widehat{X}_2$) there is only one $p_2 \in \widehat{X}_2$ (resp. $p_1 \in \widehat{X}_1$) such that $(p_1, p_2) \in X$;

As for (i), let $S_1 = \mathrm{fn}(p_1) \cap (\mathcal{N}_Fg_1 \setminus \mathcal{N}_F\widehat{g}_1)$ and suppose that S_1 is not empty. Let $x \in S_1$ and $x\sigma_1 = x'$ (for the sake of readability we write σ_1 also for the function $\mathcal{N}_F\sigma_1$). We have two cases:

1. Every $y \in \mathrm{fn}(p_2)$ is such that $y\sigma_2 \neq x'$: then x cannot be in $\mathrm{fn}(p_1)$, because otherwise we would have $p_1[\sigma_1] \neq p_2[\sigma_2]$;
2. There is $y \in \mathrm{fn}(p_2)$ such that $y\sigma_2 = x'$: then x and y stem from items i of g_1 and j of g_2, respectively, such that $\sigma_1(i) = \sigma_2(j)$, so i appears also in $\widehat{g}_1$, which implies $x \in \mathcal{N}_F\widehat{g}_1$.

Both cases imply $x \notin S_1$, which is absurd.

As for (ii), consider the following function

$$\varphi : \widehat{X}_1 \xrightarrow{[|\rho_1|]^{-1}} N_1g \xrightarrow{[|\rho_2|]} \widehat{X}_2;$$

we can let p_2 be $\varphi(p_1)$: by commutativity of (3) we have

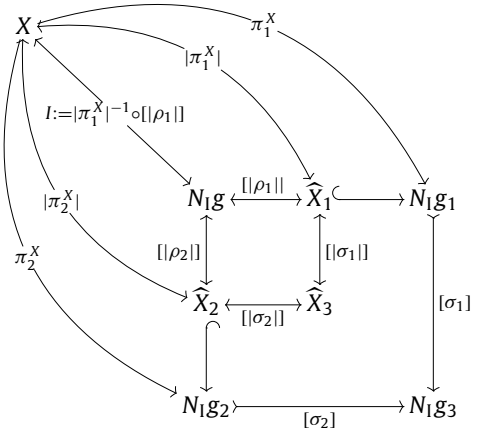
$$p_2[\sigma_2] = p_2[|\sigma_2|] = p_1[|\sigma_1|] = p_1[\sigma_1],$$

so $(p_1, p_2) \in X$. Now, suppose there is another p_2' such that $(p_1, p_2') \in X$. By definition of X, $p_1[\sigma_1] = p_2'[\sigma_2]$, but also $p_1[\sigma_1] = p_2[\sigma_2]$, so $p_2[\sigma_2] = p_2'[\sigma_2]$ and, by injectivity of $[\sigma_2]$, $p_2' = p_2$.

Now, thanks to the above observations, we have that the two functions

$$|\pi_1^X| : X \to \widehat{X}_1 \qquad |\pi_2^X| : X \to \widehat{X}_2$$

are well-defined and bijective. Therefore we have that the following diagram commutes



□

Proof of Theorem 4.12.

(i) One direction is given by Proposition 4.7, the other one by Theorem 4.9.

(ii) Given a $\mathbf{G}_I$-indexed bisimulation $\{R_g\}_{g \in |\mathbf{G}_I|}$ on $(N_I, \longrightarrow_\nu)$, we will show that

$$R^* := \bigcup_{g \in |\mathbf{G}_I|} R_g$$

is a network conscious bisimulation that is closed under injective renamings. Consider $(p, q) \in R^*$ and suppose $p, q \in N_I g$. Condition (i) of Definition 2.10 says that p and q are able to simulate each other and their continuations are again in R^*; freshness of $\mathrm{bn}(\alpha)$ w.r.t. q is guaranteed due to $\mathrm{bn}(\alpha)$ being generated by $\delta^\bullet$ or $\delta^{\bullet\to\bullet}$, thus fresh by construction. Finally, closure under injective renamings is guaranteed by (ii) of Definition 2.10.

Viceversa, consider a network conscious bisimulation R. Then is easy to see that the following family of relations is a $\mathbf{G}_I$-indexed bisimulation on $(N_I, \longrightarrow_\nu)$:

$$\{R_g := R \cap N_I g\}_{g \in \mathbf{G}_I} \qquad \square$$

Proof of Theorem 4.14. $\mathscr{E}$ and $\mathscr{R}$ are a pair of adjoint functors between accessible categories, hence are accessible themselves (Proposition 2.23 of [22]). Moreover, they are both right adjoints ($\mathscr{R}$ is right adjoint to the left Kan extension along $\mathbf{G}_I \hookrightarrow \mathbf{G}$), thus preserve limits. Therefore, being $\widehat{B}$ the composition of three accessible and weak-pullback-preserving functors, it has the same properties. □

Proof of Proposition 4.15. One direction of the mapping is given by

$$(\mathscr{R}P, \rho) \quad \longmapsto \quad (P, \rho') \qquad \rho' : P \xrightarrow{\eta_P} \mathscr{E}\mathscr{R}P \xrightarrow{\mathscr{E}\rho} \mathscr{E}B\mathscr{R}P = \widehat{B}P$$

the other one by

$$(P, \phi) \quad \longmapsto \quad (\mathscr{R}P, \phi') \qquad \phi' : \mathscr{R}P \xrightarrow{\mathscr{R}\phi} \mathscr{R}\widehat{B}P = \mathscr{R}\mathscr{E}B\mathscr{R}P \xrightarrow{\varepsilon_{B\mathscr{R}P}} B\mathscr{R}P \qquad \square$$

Proof of Proposition 17. Consider a $\widehat{B}$-coalgebra (P, ρ) and the transition relation $\longrightarrow_\rho$ given by

$$g \vdash p \xrightarrow{(\sigma,\alpha)}_\rho g' \vdash p' \iff (\alpha, p' \in Pg') \in (\rho_g(p))_\sigma .$$

We have that $(P, \longrightarrow_\rho)$ is a $\mathbf{G}_I$-IL$_{nc}$TS$_{SAT}$. In fact, requirement (i) of Definition 4.16 characterizes transitions according to $\widehat{B}$: explicitly, we have

$$\rho_g(p) \in \prod_{\sigma : g \to g'' \in \|\mathbf{G}\|} B\mathscr{R}P(g'')$$

so each label of a transition of p is of the form $(\sigma : g \to g'', \alpha)$, where $(\alpha, p') \in B\mathscr{R}P(g'')$, which means that the shape of the continuation $g' \vdash p'$ is given by (ii) of Definition 4.6. Condition (iii.a) amounts to require, for the tuple $t = \rho_g(p)$

$$(\alpha, p') \in t_\sigma \iff (\alpha[\sigma'], p[\varphi(\sigma')]) \in t_{\sigma' \circ \sigma}$$

but the pairs on the right-hand side are exactly the elements of $t_\sigma[\sigma']_{B\mathscr{R}P}$, so the requirement can be rewritten as $t_{\sigma' \circ \sigma} = t_\sigma[\sigma']_{B\mathscr{R}P}$, which is the closure under monos condition characterizing the tuples in $\widehat{B}P(g)$. Finally, condition (iii.b) reflects the naturality of ρ, explicitly

$$\begin{array}{ccc} p \in Pg & \xmapsto{\rho_g} & t \text{ s.t. } (\alpha, p') \in t_{\sigma' \circ \sigma} \\ {\scriptstyle [\sigma]_P}\downarrow & & \downarrow{\scriptstyle [\sigma]_{\widehat{B}P}} \\ p[\sigma] \in Pg' & \xmapsto[\rho_{g'}]{} & t' \text{ s.t. } (\alpha, p') \in t'_{\sigma'} \end{array} \qquad \square$$

Proof of Theorem 4.18.

(i) Analogous to (i) of Theorem 4.12.

(ii) Each **G**-indexed bisimulations $\{R_g\}_{g \in |\mathbf{G}|}$ on $(N, \longrightarrow_{\nu\text{-}SAT})$ is equivalent to a $\widehat{B}$-bisimulation R, thanks to (i); R, by Proposition 4.15 (instantiated to those B-coalgebras that are B-bisimulations) is in turn equivalent to a B-bisimulation $\mathscr{R}R$. By Theorem 4.12, this corresponds to a network conscious bisimulation, and precisely to $\bigcup_{g \in |\mathbf{G}_!|} \mathscr{R}R(g)$, which is clearly equal to $\bigcup_{g \in |\mathbf{G}|} R_g$, thus is closed under all renamings. $\square$

Proof of Proposition 5.4. By induction on the inference of $p \overset{\Lambda}{\Longrightarrow} q$. We show the most relevant case, i.e. when this transition is inferred through (COM). We have that $p \overset{\Lambda}{\Longrightarrow} q$ is inferred from

$$p \xrightarrow{\Lambda' = (R)(\bullet; W; \tilde{a}br \mid ab'x; W'; \bullet \mid \Theta)} q',$$

so $\Lambda \equiv_\Lambda (R')(\bullet; W; W'; \bullet \mid \Theta)$ and $q \equiv (R'')q'(\sigma_b \circ \sigma_r)$. By induction hypothesis Λ' and q' are well-formed. Therefore also Λ is well-formed, because it is obtained from Λ' by just removing some names, and it is simple, otherwise (COM) could not be applied.

As for $(R'')q'(\sigma_b \circ \sigma_r)$, $q'(\sigma_b \circ \sigma_r)$ is clearly well-formed, because σ_b and σ_r avoid captures. Now we show that, if $a \in R''$, then $l_{ab} \in \mathrm{fn}(q'(\sigma_b \circ \sigma_r))$ only if $l_{ab} \in R''$. Suppose, by absurd, that $l_{ab} \in \mathrm{fn}(q'(\sigma_b \circ \sigma_r))$ but $l_{ab} \notin R''$. Then we have that l_{ab} must be in $\mathrm{fn}(p)$, $\mathrm{Bn}(\Lambda)$ or $\mathrm{Obj}_{\mathrm{in}}(\Lambda)$:

- $l_{ab} \in \mathrm{fn}(p)$ contradicts $a \in R'' \subseteq \mathrm{bn}(p)$, because we assumed that free and bound names of p are distinct;
- if $l_{ab} \in \mathrm{Bn}(\Lambda)$ then either $l_{ab} \in R'$ or $(l_{(ab)})$ is the placeholder of an input path in Λ: in the first case also a must be in R', because $l_{ab} \in \mathrm{Obj}(\Theta)$ and thus $a \in \mathrm{Obj}(\Theta)$, but then $a \notin R''$, which is a contradiction; in the second case, a would be already bound in Λ, but this is not allowed by the inference rules;
- $l_{ab} \in \mathrm{Obj}_{\mathrm{in}}(\Lambda)$ contradicts $a \in R$, because otherwise l_{ab} would be bound and Λ non-well-formed.

Summarizing, either $l_{ab} \notin \mathrm{fn}(q'(\sigma_b \circ \sigma_r))$ or $l_{ab} \in \mathrm{fn}(q'(\sigma_b \circ \sigma_r))$ and $l_{ab} \in R''$, so q is well-formed. $\square$

A.1. Proof of Theorem 5.6

We introduce the following notation:

$\Pi, \Pi', \ldots, \Pi_1, \Pi_2, \ldots$:	(forrests of) proof trees
$\Pi\sigma$:	proof obtained by applying σ to every process and concurrent path in Π
$\mathrm{dom}(\sigma)$:	set of names that are not mapped to themselves by σ;
$\mathrm{img}(\sigma)$:	image of $\mathrm{dom}(\sigma)$ through σ;
$\mathrm{ds}(\alpha)$:	destination site of a path α, if any.
$x\sigma^*$:	result of applying σ to both the free and bound names of x

We call *non-linear rules* those rules that depend on the equality of certain names in the premises, namely (COM), (SRV-IN), (SRV-OUT), (SRV-SRV), which require that at least the interaction sites of two paths in the premise are equal; we call *linear rules* all the other ones.

We need a plethora of lemmata in order to prove the main result.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

Lemma A.2. *Suppose $p \stackrel{\Lambda}{\Longrightarrow} q$ and let Ω be a sequence of non-linear rule instances inferring $p \stackrel{\Lambda'}{\Longrightarrow} q'$ from this transition. Then q' is of the form $(R)q\sigma$, for some R and σ. Moreover, given any other transition $p'' \stackrel{\Lambda}{\Longrightarrow} q''$, there is a sequence of non-linear rule instances that infers $p'' \stackrel{\Lambda'}{\Longrightarrow} (R)q''\sigma$ from it.*

Proof. Non-linear rules, and in particular (COM), can only modify the continuation of the premise by adding restrictions and renaming it. This justifies the first part of the statement. The second part follows from observing that each rule instance in Ω does not look at the source process, so we can replace p with p'' and q with q'' in Ω in order to infer $p'' \stackrel{\Lambda'}{\Longrightarrow} (R)q''\sigma$ from $p'' \stackrel{\Lambda}{\Longrightarrow} q''$. □

Lemma A.3. *We can define three operations on proofs:*

(i) **renaming:** *Given $p \stackrel{\Lambda}{\Longrightarrow} q$, with proof Π, and a renaming σ such that $\Lambda\sigma$ is simple, then $p\sigma \stackrel{\Lambda\sigma}{\Longrightarrow} q\sigma$ has proof $\Pi\sigma$.*

(ii) **α-conversion**: *Given $p \stackrel{\Lambda}{\Longrightarrow} q$, with proof Π, for any $\sigma = [r'/r]$ such that $r \in \mathrm{Bn}(\Lambda)$ and r' is fresh w.r.t. p and Λ, we have that $p\sigma^* \stackrel{\Lambda\sigma^*}{\Longrightarrow} q\sigma$ has proof $\Pi\sigma^*$.*

(iii) **Input object restriction**: *Given $p \stackrel{(R)(abr;W;\bullet \mid \Theta)}{\Longrightarrow} q$, with proof Π, suppose $a(s).p' \stackrel{aar;\bullet}{\Longrightarrow} p'[r/s]$ is the axiom for the input path in Π. Then, for any fresh s', there is transition*

$$p[s'/s]^* \stackrel{(R)(ab(s');W;\bullet \mid \Theta)}{\Longrightarrow} q'$$

with proof Π' obtained as follows:
(a) *r is replaced with (s') in each input path from which $abr; W; \bullet$ is constructed;*
(b) *$p'[r/s]$ is replaced with $p'[s'/s]$ throughout the proof.*
Moreover, $q'[r/s'] = q$.

Proof. All the three statements can be proved by induction on the depth of Π.

(i) By cases on the type of the last rule instance in Π; we show some relevant ones:

Case (OPEN) Then $p = (r'')p'$, $\Lambda \equiv_\Lambda (r'')(\Lambda' /\!\!/ r'')$ and the transition is inferred from $p \stackrel{\Lambda'}{\Longrightarrow} q$. Suppose this last transition has proof Π'. $\Lambda'\sigma$ is simple because, if r'' is a link, σ does not map any other link to it by Convention A.1, so the induction hypothesis can be applied, yielding $p'\sigma \stackrel{\Lambda'\sigma}{\Longrightarrow} q\sigma$. Using (OPEN) again, whose side condition holds because σ does not map any other free name to r'', we get the thesis.

Case (PAR) Then $p = p_1 \mid p_2$, $\Lambda \equiv_\Lambda \Lambda_1 \mid \Lambda_2$ and the transition is inferred from $p_1 \stackrel{\Lambda_1}{\Longrightarrow} q_1$ and $p_2 \stackrel{\Lambda_2}{\Longrightarrow} q_2$, so $q = q_1 \mid q_2$. Suppose these transitions have proofs Π_1 and Π_2. Clearly, if $(\Lambda_1 \mid \Lambda_2)\sigma$ is simple, so are $\Lambda_1\sigma$ and $\Lambda_2\sigma$, thus, by induction hypothesis, $p_1\sigma \stackrel{\Lambda_1\sigma}{\Longrightarrow} q_1\sigma$ and $p_2\sigma \stackrel{\Lambda_2\sigma}{\Longrightarrow} q_2\sigma$ have proofs $\Pi_1\sigma$ and $\Pi_2\sigma$, respectively. Moreover, by Convention A.1, σ cannot break (PAR) side conditions, so, applying (PAR) again, we get the thesis.

Case (COM) Then the transition is inferred from $p \stackrel{(R)(\alpha_1 \mid \alpha_2 \mid \Theta)}{\Longrightarrow} q$, with $\alpha_1 = \bullet; W; \bar{a}br$ and $\alpha_2 = ab'x; W'; \bullet$, so $\Lambda \equiv_\Lambda (R')(\bullet; W; W'; \bullet \mid \Theta)$ and $q = (R'')q'\sigma'$. Suppose the premise has proof Π'. If $(R')(\bullet; W; W'; \bullet \mid \Theta)\sigma$ is simple, thus $W\sigma$ and $W'\sigma$ are simple, so are $\alpha_1\sigma$ and $\alpha_2\sigma$, hence, by induction hypothesis, $p\sigma \stackrel{(R)(\alpha_1 \mid \alpha_2 \mid \Theta)\sigma}{\Longrightarrow} q'\sigma$ has proof $\Pi'\sigma$. Suppose $\sigma' = [r_1/r'_1, \ldots, r_k/r'_k]$. Applying (COM) to the renamed transition we get

$$p\sigma \stackrel{(R')(\bullet;W;W';\bullet \mid \Theta)\sigma}{\Longrightarrow} (R''\sigma)q'([r_1\sigma/r'_1, \ldots, r_k\sigma/r'_k] \circ \sigma),$$

but the continuation coincides with $(R''\sigma)q'(\sigma \circ \sigma')$. The thesis follows.

(ii) A straightforward adaptation of the proof of (i). The cases (RES) and (OPEN) are treated by applying the renaming operation (i) to the premises with $\sigma = [r'/r]$ whenever $p = (r)p'$. Notice that σ^* cannot make paths non-simple, because r and r' are both fresh w.r.t. the lists of links appearing in labels.

(iii) By cases on the type of the last rule of Π. We show two cases: the first clarifies the most, the second exemplifies the inductive step.

Case (IN): Then $p = a(s).p'$, Λ is just $aar; \bullet$, $q = p'[r/s]$ and Π is an axiom. Then Π' is

$$a(s').p'[s'/s] \stackrel{aa(s);\bullet}{\Longrightarrow} p'[s'/s]$$

which clearly satisfies (a) and (b). Finally, we have

$$(p'[s'/s])[r/s'] = p'[r/s] = q.$$

Case (OPEN): Then $p = (r')p'$, $\Lambda \equiv_\Lambda (r')(R)(abr; W; \bullet /\!/ r' \mid \Theta /\!/ r')$ and the transition is inferred from $p' \xrightarrow{(R)(abr;W;\bullet \mid \Theta)} q$. Suppose this last transition has proof Π''. Clearly it contains the same input axiom as Π. So, by induction hypothesis, there is

$$p'[s'/s]^* \xrightarrow{(R)(ab(s');W;\bullet \mid \Theta)} q'$$

where q' is q after (b) is applied, and this transition has proof Π''', obtained via (a) and (b). We can apply (OPEN) again and get the thesis. □

Lemma A.4. *Suppose $p \xrightarrow{\Lambda} q$ has a proof with the following structure*

$$\dfrac{\begin{array}{c}\Pi \\ \text{non-linear rule } \rho_1 \\ \text{linear rule } \rho_2\end{array}}{p \xrightarrow{\Lambda} q}$$

and let α_1 and α_2 be the paths concatenated by ρ_1. Suppose $\mathrm{is}(\alpha_1) \cap \mathrm{is}(\alpha_2) \cap \mathrm{bn}(p) = \varnothing$. Then there is Π', a linear rule instance ρ_2', a non-linear rule instance ρ_1', $p^* \equiv p$ and q^*, obtained by bringing some restrictions of q to the top level, such that

$$\dfrac{\begin{array}{c}\Pi' \\ \rho_2' \\ \rho_1'\end{array}}{p^* \xrightarrow{\Lambda} q^*}$$

Proof. We have to consider all the combinations of ρ_1 and ρ_2 where the objects of α_1 and α_2 are involved in the side conditions of ρ_2. If they are not, we can simply put $\rho_1' = \rho_2$, $\rho_2' = \rho_1$ and $\Pi' = \Pi$. The only non-trivial cases are when ρ_2 is (RES), (OPEN) or (PAR). We show the proof of the statement for ρ_1 being (COM), which is the most involved case.

Case $\rho_2 =$ (RES): Then $p = (r')p'$ and the transition is inferred as follows

$$(\text{RES})\ \dfrac{(\text{COM})\ \dfrac{\begin{array}{c}\Pi \\ p' \xrightarrow{(R)(\bullet;W;\bar{a}br \mid ab'x;W';\bullet \mid \Theta)} q'\end{array}}{p' \xrightarrow{(R')(\bullet;W;W';\bullet \mid \Theta)} (R'')q'(\sigma_b \circ \sigma_r)}}{(r')p' \xrightarrow{\Lambda \equiv_\Lambda (R')(\bullet;W/r';W'/r';\bullet \mid \Theta/r')} (r')(R'')q'(\sigma_b \circ \sigma_r)}$$

We want (COM) to appear after the rule treating the restriction of r'. We have two cases:

- $r' \in \mathrm{n}(r)$: if $x = r$ then the earlier restriction of r' should be treated through (OPEN). However, we need to make the input object in the top transition bound in order to get a proper premise of (OPEN). Let s be the input placeholder in p' for $ab'x; W'; \bullet$ and let s' be fresh. Then we can apply (iii) of Lemma A.3 and get

$$p'[s'/s]^* \xrightarrow{(R)(\bullet;W;\bar{a}br \mid ab'(s');W';\bullet \mid \Theta)} q'';$$

where $q''[r/s'] = q'$. This transition has proof Π', obtained as described in (iii) of Lemma A.3. We can put $\rho_2' =$ (OPEN), because (OPEN) side condition is satisfied due to (RES)'s one and $r' \neq a$ by hypothesis, and $\rho_1' =$ (COM). So we have

$$(\text{COM})\ \dfrac{(\text{OPEN})\ \dfrac{\begin{array}{c}\Pi' \\ p'[s'/s]^* \xrightarrow{(R)(\bullet;W;\bar{a}br \mid ab'(s');W';\bullet \mid \Theta)} q''\end{array}}{(r')p'[s'/s]^* \xrightarrow{(r')(R)(\bullet;W/r';\bar{a}br \mid ab'(s');W'/r';\bullet \mid \Theta/r')} q''}}{(r')p'[s'/s]^* \xrightarrow{(R')(\bullet;W/r';W'/r';\bullet \mid \Theta/r')} (r')(R'')q''([r/s'] \circ \sigma_b) = (r')(R'')q''\sigma_b}$$

Now, since $x = r$ implies $\sigma_r = id$, the continuation of the inferred transition is indeed the one of the original proof. Finally, the case $x = (s)$ is trivial: we can permute the rules without affecting Π.

- $r' = b \notin \mathrm{n}(r)$: in this case necessarily $b \notin R$, so $b = b'$ and $\sigma_b = id$. We simply have $\Pi' = \Pi$, $\rho_2' =$ (COM) and $\rho_1' =$ (OPEN).

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

Case $\rho_2 = (\textsc{open})$**:** then $p = (r')p'$ and the transition is inferred as follows

$$(\textsc{open})\ \frac{(\textsc{com})\ \dfrac{p' \xrightarrow{(R)(\bullet;W;\bar{a}br \mid ab'x;W';\bullet \mid \Theta)} q'}{p' \xrightarrow{(R')(\bullet;W;W';\bullet \mid \Theta)} (R'')q'(\sigma_b \circ \sigma_r)}}{(r')p' \xrightarrow{\Lambda \equiv_\Lambda (r')(R')(\bullet;W/r';W'/r';\bullet \mid \Theta/r')} (R'')q'(\sigma_b \circ \sigma_r)}$$

The proof then is a straightforward adaptation of the previous case.

Case $\rho_2 = (\textsc{par})$**:** Then $p = p_1 \mid p_2$ and the transition is inferred as follows

$$(\textsc{par})\ \frac{(\textsc{com})\ \dfrac{p_1 \xrightarrow{(R)(\bullet;W;\bar{a}br \mid ab'x;W';\bullet \mid \Theta)} q_1}{p_1 \xrightarrow{(R')(\bullet;W;W';\bullet \mid \Theta)} (R'')q_1\sigma} \qquad p_2 \xrightarrow{\Lambda_2} q_2}{p_1 \mid p_2 \xrightarrow{(R')(\bullet;W;W';\bullet \mid \Theta) \mid \Lambda_2} (R'')q_1\sigma \mid q_2}$$

Now, we would like to permute (com) and (par), but there may be some bound names in $\{a, b, b'\} \cup \mathrm{n}(x) \cup \mathrm{n}(r)$ that conflict with Λ_2, breaking (par) side conditions. So we have to repeatedly apply (ii) of Lemma A.3 to α-convert Π_1. Let σ^*_α be the α-converting renaming. We can build the following proof

$$(\textsc{par})\ \frac{(\textsc{com})\ \dfrac{p_1\sigma^*_\alpha \xrightarrow{(R\sigma^*_\alpha)(\bullet;W;\overline{a\sigma_\alpha}b\sigma_\alpha r\sigma_\alpha \mid a\sigma_\alpha b'\sigma_\alpha x\sigma_\alpha;W';\bullet \mid \Theta)} q_1\sigma_\alpha \qquad p_2 \xrightarrow{\Lambda_2} q_2}{p_1\sigma^*_\alpha \mid p_2 \xrightarrow{(R\sigma^*_\alpha)(\bullet;W;\overline{a\sigma_\alpha}b\sigma_\alpha r\sigma_\alpha \mid a\sigma_\alpha b'\sigma_\alpha x\sigma_\alpha;W';\bullet \mid \Theta) \mid \Lambda_2} q_1\sigma_\alpha \mid q_2}}{p_1\sigma^*_\alpha \mid p_2 \xrightarrow{(R')(\bullet;W;W';\bullet \mid \Theta) \mid \Lambda_2} (R''\sigma^*_\alpha)(q_1\sigma_\alpha \mid q_2)\sigma'}$$

where clearly $p_1\sigma^*_\alpha \mid p_2 \equiv p_1 \mid p_2$ and σ' maps the α-converted extruded names to the α-converted input placeholders, so, since $\mathrm{dom}(\sigma')$ is fresh w.r.t. q_2, α-conversion allows us to recover the original mapping, namely $(R''\sigma_\alpha)(q_1\sigma_\alpha \mid q_2)\sigma' \equiv (R'')(q_1\sigma \mid q_2)$. Notice that the scope of R'' now includes both $q_1\sigma$ and q_2. □

Lemma A.5. *Given a process p and a renaming σ, if $p\sigma \xrightarrow{\Lambda} q$ has a proof Π such that every instance of a non-linear rule in Π whose premise is*

$$p\sigma \xrightarrow{(R)(\alpha_1 \mid \alpha_2 \mid \Theta)} q'$$

where α_1 and α_2 are the paths concatenated in the consequence, is such that

$$\mathrm{obj}(\alpha_1) \cup (\mathrm{is}(\alpha_1) \cap \mathrm{is}(\alpha_2)) = \varnothing,$$

where the intersection is taken over $\mathrm{img}(\sigma)$, then $p \xrightarrow{\Lambda'} q'$, where $\Lambda = \Lambda'\sigma$ and $q = q'\sigma$.

Proof. By induction on the depth of Π. Suppose Π has depth at least one (the base cases are trivial), the proof proceeds by cases on the last rule of Π. We show two of them, the other ones are analogous:

Case $(\textsc{open})$**:** Then $p = (r'')p'$, $\Lambda \equiv_\Lambda (r'')(\Lambda'' /\!\!/ r)$ and the transition is inferred as follows

$$(\textsc{open})\ \frac{p'\sigma \xrightarrow{\Lambda''} q}{(r'')p'\sigma \xrightarrow{(r'')(\Lambda'' /\!\!/ r)} q}$$

By induction hypothesis we have $p' \xrightarrow{\widetilde{\Lambda}} \widetilde{q}$, where $\Lambda'' = \widetilde{\Lambda}\sigma$ and $q = \widetilde{q}\sigma$. We can apply (open) to this transition, because σ does not affect r'' and, consequently, the rule's side condition. This yields $(r'')p' \xrightarrow{(r'')(\widetilde{\Lambda} /\!\!/ r'')} \widetilde{q}$, which satisfies the statement.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

Case (COM): Then $\Lambda \equiv_\Lambda (R')\,(\bullet; W; W'; \bullet \mid \Theta)$, $q = (R'')q'\sigma$ and the transition is inferred from

$$(\text{COM})\ \frac{\Pi'}{p\sigma \xrightarrow{(R)\ (\bullet; W; \bar{a}br'' \mid ab'x; W'; \bullet \mid \Theta)} q'} \qquad p\sigma \xrightarrow{(R')\ (\bullet; W; W'; \bullet \mid \Theta)} (R'')q'\sigma'$$

By induction hypothesis we have $p \xrightarrow{(\widetilde{R})\ (\widetilde{\alpha}_1 \mid \widetilde{\alpha}_2 \mid \widetilde{\Theta})} \widetilde{q}$ where $\widetilde{\alpha}_1\sigma = \bullet; W; \bar{a}br''$, $\widetilde{\alpha}_2\sigma = ab'x; W'; \bullet$, $\widetilde{R}\sigma = R$, $\widetilde{\Theta}\sigma = \Theta$ and $\widetilde{q}\sigma = q'$. By hypothesis, σ acts as the identity on a, b and r'', so we have: $\text{is}(\widetilde{\alpha}_1) = \text{is}(\widetilde{\alpha}_2) = a$; $\text{obj}(\widetilde{\alpha}_1) = \text{obj}(\widetilde{\alpha}_2)$ whenever $r'' \notin R$; $\text{ds}(\widetilde{\alpha}_1) = \text{ds}(\widetilde{\alpha}_2) = b$ whenever $b \notin R$. Therefore we can apply (COM) again and get the desired transition. The continuation of such transition is indeed renamed through σ': if some names among r'',x, b and b' are bound in $p\sigma$, they are also bound in p, because σ did not affect them by Convention A.1. □

Lemma A.6. *Given a process p and a renaming σ, suppose $p\sigma \xRightarrow{\Lambda} q$ has proof Π. Then there is a transition $p \xRightarrow{\Lambda'} q'$ and a sequence Ω of non-linear rule instances such that from $p\sigma \xRightarrow{\Lambda'\sigma} q'\sigma$, after applying Ω, we get $p\sigma \xRightarrow{\Lambda} q^*$, where q^* is q with some unguarded restrictions brought to the top level.*

Proof. Since, by Convention A.1, $\text{img}(\sigma) \cap \text{bn}(p) = \varnothing$, we can repeatedly use Lemma A.4 to push towards the root of the proof all the instances of non-linear rules in Π that concatenate α_1 and α_2 such that $\text{obj}(\alpha_1)$ or $\text{is}(\alpha_1) \cap \text{is}(\alpha_2)$ are in the image of σ, i.e. those that violate the hypothesis of Lemma A.5. More precisely:

1. Take the deepest subproof $\widetilde{\Pi}$ of Π such that Lemma A.4 can be applied; let $\widetilde{p} \xRightarrow{\widetilde{\Lambda}} \widetilde{q}$ be its inferred transition.
2. Permute the last two rule instances in $\widetilde{\Pi}$ using Lemma A.4, let $\widetilde{\Pi}'$ the resulting proof and $\widetilde{p}^* \xRightarrow{\widetilde{\Lambda}} \widetilde{q}^*$ its inferred transition.
3. Replace $\widetilde{\Pi}$ with $\widetilde{\Pi}'$ in Π and each occurrence of $\widetilde{p}$ and $\widetilde{q}$ with $\widetilde{q}^*$ and $\widetilde{q}^*$, respectively, in the piece of proof below $\widetilde{\Pi}$. The resulting proof is indeed valid, because step 1 does not affect the label.

Repeat these steps until obtaining a new proof for $p^*\sigma \xRightarrow{\Lambda} q^*$. This proof has an upper part Π'' and a lower part Ω', the latter containing all the non-linear rules that violate the hypothesis of Lemma A.5 when considering the renaming σ. Now, suppose Π'' infers $p^*\sigma \xRightarrow{\widehat{\Lambda}} \widehat{q}$. We can apply Lemma A.5 to this transition and get the required $p^*\sigma\,(\equiv p) \xRightarrow{\Lambda'} q'$ such that $\widehat{\Lambda} = \Lambda'\sigma$ and $\widehat{q} = q\sigma$. Finally, Lemma A.2 allows us to construct the required Ω out of Ω'. □

Lemma A.7. *The relation $\mathcal{R} = \bigcup_{n\in\omega} \mathcal{R}_n$, where*

$$\begin{aligned}
\mathcal{R}^1 &= \{((r)p_1 \mid p_2, (r)(p_1 \mid p_2)) \mid r \notin \text{fn}(p_2)\} \\
\mathcal{R}^2 &= \{(p_1 \mid p_2, p_2 \mid p_1)\} \\
\mathcal{R}^3_{n+1} &= \{((r)p, (r)q) \mid (p, q) \in \mathcal{R}_n\} \\
\mathcal{R}^4_{n+1} &= \{(p\sigma, q\sigma) \mid (p, q) \in \mathcal{R}_n\}
\end{aligned}$$

$$\mathcal{R}_0 = \sim^{NC}_{\kappa} \cup \mathcal{R}^1 \cup \mathcal{R}^2 \qquad \mathcal{R}_{n+1} = \mathcal{R}^3_{n+1} \cup \mathcal{R}^4_{n+1} \cup \mathcal{R}_n$$

is a network conscious bisimulation.

Proof. Given $(p, q) \in \mathcal{R}$, we have to prove that p and q can simulate each other's transitions and that their continuations are related by $\mathcal{R}$. To do this, we proceed by induction on n, considering any $p' \equiv p$ and $q' \equiv q$ such that $(p', q') \in \mathcal{R}_n$. For the base case, we treat separately the case of (p', q') being in $\mathcal{R}^1$ and $\mathcal{R}^2$. For the inductive step, we do the same, considering R^3_{n+1} and R^4_{n+1}, for $n \geqslant 0$. Considering processes that are structurally congruent to the original ones is harmless, because $\equiv \subseteq \sim^{NC}_{\kappa} \subseteq \mathcal{R}$, and allows us to ignore transitions that are not inferred directly through the rules.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

$p \equiv (r)p_1 \mid p_2$, $q \equiv (r)(p_1 \mid p_2)$ Suppose $((r)p_1 \mid p_2, (r)(p_1 \mid p_2)) \in \mathcal{R}^1$ and consider a transition $(r)p_1 \mid p_2 \xRightarrow{\Lambda} p'$. We prove that $(r)(p_1 \mid p_2)$ can simulate this transition by cases on the last rule used to infer it.

Case (PAR): Suppose it is inferred as follows

$$\dfrac{\dfrac{p_1 \xRightarrow{\Lambda_1} q_1}{(r)p_1 \xRightarrow{(r)(\Lambda_1 /\!\!/ r)} p_1'} \quad \vdots\ \Omega\ \vdots \quad (r)p_1 \xRightarrow{\Lambda_1'} (R)p_1'\sigma \quad \vdots\ \Omega_{(r)}\ \vdots \quad (r)p_1 \xRightarrow{\Lambda_1''} (R')(R)p_1'(\sigma_{(r)} \circ \sigma) \qquad p_2 \xRightarrow{\Lambda_2} p_2'}{(r)p_1 \mid p_2 \xRightarrow{\Lambda_1'' \mid \Lambda_2} (R')(R)p_1'(\sigma_{(r)} \circ \sigma) \mid p_2'}\ (\text{PAR})$$

where Ω and $\Omega_{(r)}$ are sequences of non-linear rule instances (the rule on top may be (RES): this case is analogous). In particular, suppose $\Omega_{(r)}$ contains those rules that treat the communication of (r), if any.

The idea is to build a new proof where (PAR) is applied before (OPEN). However, some names in $\mathrm{Bn}(\Lambda_1) \setminus \mathrm{Bn}(\Lambda_1'')$ and $\mathrm{n}(r)$ that are also in $\mathrm{Fn}(\Lambda_2)$ may prevent the application of these rules. So we use (i) and (ii) of Lemma A.3 to replace those names with fresh ones. Let $\widehat{\sigma}$ be the mapping that performs this operation. The new proof is

$$\dfrac{\dfrac{\dfrac{p_1\widehat{\sigma}^* \xRightarrow{\Lambda_1\widehat{\sigma}^*} p_1'\widehat{\sigma} \qquad p_2 \xRightarrow{\Lambda_2} p_2}{p_1\widehat{\sigma}^* \mid p_2 \xRightarrow{\Lambda_1\widehat{\sigma}^* \mid \Lambda_2} p_1'\widehat{\sigma} \mid p_2'} \quad \vdots\ \Omega'\ \vdots \quad p_1\widehat{\sigma}^* \mid p_2 \xRightarrow{\Lambda_1'\widehat{\sigma}^* \mid \Lambda_2} (R\widehat{\sigma})(p_1'\widehat{\sigma} \mid p_2')\sigma'}{(\widehat{r})(p_1\widehat{\sigma}^* \mid p_2) \xRightarrow{(\widehat{r})(\Lambda_1'\widehat{\sigma}^* /\!\!/ \widehat{r} \mid \Lambda_2)} (R\widehat{\sigma})(p_1'\widehat{\sigma} \mid p_2')\sigma'}\ (\text{OPEN}) \quad \vdots\ \Omega'_{(\widehat{r})}\ \vdots}{(\widehat{r})(p_1\widehat{\sigma}^* \mid p_2) \xRightarrow{\Lambda_1'' \mid \Lambda_2} (R'\widehat{\sigma})(R\widehat{\sigma})(p_1'\widehat{\sigma} \mid p_2')\sigma'_{(\widehat{r})} \circ \sigma'}\ (\text{PAR})$$

where the rule instances occurring in Ω' and $\Omega'_{(r)}$ are of the same kind of those in Ω and $\Omega_{(r)}$, concatenate the same pairs of paths (modulo α-conversion of bound names), but involve different processes. This does not affect the applicability of the rules.

$(\widehat{r})\Lambda_1'\widehat{\sigma}^* /\!\!/ \widehat{r}$ can be thought of as the α-converted version of Λ_1', but the α-converted bound names, namely those in $R'\widehat{\sigma}$, are all removed by the subsequent non-linear rules. In particular, $\widehat{r}$ is removed whenever $r \in \mathrm{Fn}(\Lambda_2)$, because otherwise (PAR) could not be applied in the original proof, and is equal to r whenever $r \notin \mathrm{Fn}(\Lambda_2)$. This is why we have Λ_1'' in the bottom transition.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

Clearly we have $(\widehat{r})(p_1\widehat{\sigma}^* \mid p_2) \equiv (r)(p_1 \mid p_2)$ and

$$\begin{aligned}(R'\widehat{\sigma})(R\widehat{\sigma})(p_1'\widehat{\sigma} \mid p_2')\sigma'_{(r)} \circ \sigma' &= (R'\widehat{\sigma})(R\widehat{\sigma})(p_1'(\sigma'_{(r)} \circ \sigma' \circ \widehat{\sigma}) \mid p_2'(\sigma'_{(r)} \circ \sigma')) \\ &= (R'\widehat{\sigma})(R\widehat{\sigma})(p_1'(\sigma'_{(r)} \circ \sigma' \circ \widehat{\sigma}) \mid p_2') \\ &\equiv (R')(R)(p_1'(\sigma_{(r)} \circ \sigma) \mid p_2')\end{aligned}$$

where the second equation holds due to freshness of $\mathrm{dom}(\sigma'_{(r)} \circ \sigma')$ w.r.t. p_2' and the last congruence holds because $\widehat{\sigma}$ only affects names that are either being extruded or bound input placeholders in Λ_1, so $\sigma'_{(r)} \circ \sigma' \circ \widehat{\sigma}$ replaces the α-converted input placeholders with α-converted extruded names, but these are again bound by $(R\widehat{\sigma})$ and $(R'\widehat{\sigma})$ in the continuation. The overall effect is the α-conversion of $(R')(R)(p_1'(\sigma_{(r)} \circ \sigma) \mid p_2')$.

Since $\mathcal{R}$ is closed under α-conversion and scope extension, we have

$$(\,(R'\widehat{\sigma})(R\widehat{\sigma})(p_1'\widehat{\sigma} \mid p_2')\sigma'_{(r)} \circ \sigma',\, (R')(R)p_1'(\sigma_{(r)} \circ \sigma) \mid p_2'\,) \in \mathcal{R}.$$

Case non-linear rule: The last part of the derivation of $(r)p_1 \mid p_2 \stackrel{\Lambda}{\Longrightarrow} p'$ is a sequence of non-linear rule instances Ω with an occurrence of (PAR) on top. By the previous case, the transition of $(r)p_1 \mid p_2$ inferred using (PAR) can be simulated by $(r)(p_1 \mid p_2)$, and the continuations are related by $\mathcal{R}$. By Lemma A.2, there is sequence Ω' of non-linear rule instances such that Ω and Ω' add the same restrictions and apply the same renaming to the continuations of $(r)p_1 \mid p_2$ and $(r)(p_1 \mid p_2)$ as inferred by (PAR). The thesis comes from $\mathcal{R}$ being closed under renamings and addition of restrictions.

Now consider a transition $(r)(p_1 \mid p_2) \stackrel{\Lambda}{\Longrightarrow} p'$. Again, we proceed by cases on the last rule used to infer it:

Case (OPEN): Suppose it is inferred as follows

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\mathrm{PAR})}\ p_1 \stackrel{\Lambda_1}{\Longrightarrow} p_1' \qquad p_2 \stackrel{\Lambda_2}{\Longrightarrow} p_2'}{p_1 \mid p_2 \stackrel{\Lambda_1 \mid \Lambda_2}{\Longrightarrow} p_1' \mid p_2'}}{\vdots\ \Omega_1\ \vdots}}{p_1 \mid p_2 \stackrel{\Lambda_1' \mid \Lambda_2}{\Longrightarrow} (R_1)(p_1' \mid p_2')\sigma_1}}{\vdots\ \Omega_2\ \vdots}}{(\mathrm{OPEN})\ p_1 \mid p_2 \stackrel{\Lambda'}{\Longrightarrow} (R')(R_1)(p_1 \mid p_2)\sigma \circ \sigma_1 \;\Big/\; (r)(p_1 \mid p_2) \stackrel{(r)(\Lambda' /\!\!/ r)}{\Longrightarrow} (R')(R_1)(p_1' \mid p_2')\sigma \circ \sigma_1}$$

where Ω_1 contains instances of non-linear rules that only act on Λ_1 and Ω_2 all the other ones.

Now we want to move (OPEN) before (PAR), but we have to take care of the following situation: Ω_1 and Ω_2 may contain some occurrences of (COM) concatenating α_1 and α_2 such that $\alpha_1 \in \Lambda_1$, $\alpha_2 \in \Lambda_2$ and $r \in \mathrm{obj}_{\mathrm{in}}(\alpha_2)$. Since we have to restrict r earlier in the proof, r in each α_1 will become bound, so we must turn each α_2 into a bound input path by using (iii) of Lemma A.3 in order for (COM) to be applied. Suppose Λ_2 contains n such input paths and $s_1, \ldots, s_n$ are the placeholders of the associated input prefixes. Let $s_1', \ldots, s_n'$ be fresh names and $\sigma_2 := [s_1'/s_1, \ldots, s_n'/s_n]$. By repeatedly applying (iii) of Lemma A.3 we get a transition $p_2\sigma_2^* \stackrel{\Lambda_2'}{\Longrightarrow} p_2''$ such that $p_2''[r/s_1', \ldots, r/s_n'] = p_2'$, where Λ_2' is obtained by replacing each $a_ib_ir_i; W_i; \bullet \in \Lambda_2$ with $a_ib_i(s_i); W_i; \bullet$, for $i = 1, \ldots, n$. The new proof is

$$
\begin{array}{c}
p_1 \xrightarrow{\Lambda_1} p_1' \\
\vdots \\
\Omega_1' \\
\vdots \\
(\textsc{open})\ \dfrac{p_1 \xrightarrow{\Lambda_1'} (R_1)p_1'\sigma_1}{(r)p_1 \xrightarrow{(r)(\Lambda_1' /\!\!/ r)} (R_1)p_1'\sigma_1} \qquad p_2 \xrightarrow{\Lambda_2'} p_2'' \\
(\textsc{par})\ \dfrac{}{(r)p_1 \mid p_2 \xrightarrow{(r)(\Lambda_1' /\!\!/ r) \mid \Lambda_2'} (R_1)p_1'\sigma_1 \mid p_2''} \\
\vdots \\
\Omega_2' \\
\vdots \\
(r)p_1 \mid p_2 \xrightarrow{(r)(\Lambda' /\!\!/ r)} (R')((R_1)p_1'\sigma_1 \mid p_2'')\sigma'
\end{array}
$$

where Ω_1' and Ω_2' contain instances of the same kind of rules, and concatenate the same pairs of paths, as Ω_1 and Ω_2, with the exception of input paths of the form $abr; W; \bullet$ in Ω_2', which are replaced by bound input paths of the form $ab(s); W; \bullet$. Therefore we have $\sigma' = \sigma \circ \sigma_2$ and

$$
\begin{aligned}
(R')((R_1)p_1'\sigma_1 \mid p_2'')\sigma' &= (R')((R_1)p_1'\sigma_1 \mid p_2'')\sigma \circ \sigma_2 \\
&= (R')((R_1)p_1'\sigma_1 \mid p_2''\sigma_2)\sigma && \text{by freshness of } \mathrm{dom}(\sigma_2) \text{ w.r.t. } p_1 \\
&= (R')((R_1)p_1'\sigma_1 \mid p_2')\sigma && \text{(iii) of Lemma A.3} \\
&= (R')((R_1)p_1' \mid p_2')\sigma \circ \sigma_1 && \text{by freshness of } \mathrm{dom}(\sigma_1) \text{ w.r.t. } p_2
\end{aligned}
$$

and, by closure under scope extension and renamings

$$((R')(R_1)(p_1' \mid p_2')\sigma \circ \sigma_1, (R')((R_1)p_1' \mid p_2')\sigma \circ \sigma_1) \in \mathcal{R}.$$

Case (RES): Analogous.
Non-linear rule: As before.

$p \equiv p_1 \mid p_2, q \equiv p_2 \mid p_1$ Suppose $(p_1 \mid p_2, p_2 \mid p_1) \in \mathcal{R}^2$ and $p_1 \mid p_2 \xrightarrow{\Lambda} p'$. This transition can be inferred through (PAR) or a non-linear rule. In the first case, $p_2 \mid p_1$ is able to simulate it, by the commutativity of the parallel operator of concurrent paths. Its continuation is clearly paired with p' in $\mathcal{R}$. The second case is analogous to the "non-linear rule" cases shown above.

$p \equiv (r)p', q \equiv (r)q'$ Suppose $((r)p', (r)q') \in \mathcal{R}^3_{n+1}$ and $(r)p' \xrightarrow{\Lambda} p''$, with $\mathrm{Bn}(\Lambda) \# (r)q'$. We have to prove that $(r)q'$ can simulate this transition. We proceed by cases on the last rule used to infer it.

Case (OPEN): then the transition is inferred from $p' \xrightarrow{\Lambda'} p''$ and $\Lambda \equiv_\Lambda (r)(\Lambda'/r)$. Since $(p', q') \in \mathcal{R}_n$ and clearly $\mathrm{Bn}(\Lambda') \# q'$, by induction hypothesis $q' \xrightarrow{\Lambda'} q''$, with $(p'', q'') \in \mathcal{R}$, so we can apply (OPEN) to get $(r)q' \xrightarrow{\Lambda} q''$.
Case (RES): analogous to (OPEN); the main difference is that the two transitions have $(r)p''$ and $(r)q''$ as continuations, which are paired in $\mathcal{R}$ by its closure under addition of restrictions.
Case non-linear rule: Analogous to the "non-linear rule" cases above.

$p \equiv p'\sigma, q \equiv q'\sigma$ Suppose $(p'\sigma, q'\sigma) \in \mathcal{R}^4_{n+1}$ and $p'\sigma \xrightarrow{\Lambda} p''$, with $\mathrm{Bn}(\Lambda) \# q'\sigma$. We can safely assume that

$$\mathrm{bn}(q'\sigma) \text{ are fresh w.r.t. } p'\sigma \text{ and viceversa} \tag{4}$$

(we can obtain this by α-conversion, which is contained in the bisimilarity). By Lemma A.6 and A.2 there is a transition $p' \xrightarrow{\widetilde{\Lambda}} \widetilde{p}$ and a sequence Ω of non-linear rule instances such that, if we apply Ω to $p'\sigma \xrightarrow{\widetilde{\Lambda}\sigma} \widetilde{p}\sigma$, we get $p'\sigma \xrightarrow{\Lambda} (R)(\widetilde{p}\sigma)\sigma'$,

for some R and σ' according to Lemma A.2, where the continuation is p'' with some restrictions brought to the top level. In order to apply the induction hypothesis (i.e. that $\mathcal{R}_n$ is a bisimulation) to $p' \stackrel{\widetilde{\Lambda}}{\Longrightarrow} \widetilde{p}$, we have to show that $\mathrm{Bn}(\widetilde{\Lambda}) \# q'$: this comes by observing that σ does not affect bound names, so we have $\mathrm{bn}(p') \# q'$ by (4), which holds in particular for $\mathrm{Bn}(\widetilde{\Lambda})$ (since $\mathrm{Bn}(\widetilde{\Lambda}) \subseteq \mathrm{bn}(p')$).

Therefore $q' \stackrel{\widetilde{\Lambda}}{\Longrightarrow} \widetilde{q}$, with $(\widetilde{p}, \widetilde{q}) \in \mathcal{R}$, which can be renamed to $q'\sigma \stackrel{\widetilde{\Lambda}\sigma}{\Longrightarrow} \widetilde{q}\sigma$, by (i) of Lemma A.3. By Lemma A.2 there are non-linear rules that act on this transitions by applying the same renaming and restricting the same names as Ω, yielding $q'\sigma \stackrel{\Lambda}{\Longrightarrow} (R)(\widetilde{q}\sigma)\sigma'$. By closure under renamings and addition of restrictions we have $((R)(\widetilde{p}\sigma)\sigma', (R)(\widetilde{q}\sigma)\sigma') \in \mathcal{R}$ and, by closure under scope extension, we also have $(p'', (R)(\widetilde{p}\sigma)\sigma') \in \mathcal{R}$. Finally, by transitivity we can conclude $(p'', (R)(\widetilde{q}\sigma)\sigma') \in \mathcal{R}$. □

Proof of Theorem 5.6. By cases, considering all the operators. We use the same proof technique for each operator op: we consider a relation of the form

$$\mathcal{R} = \{(op(p), op(q)) \mid p \sim_\kappa^{NC} q\} \cup \sim_\kappa^{NC}$$

and we show that it is a bisimulation.

Case op $\in \{\bar{\mathbf{a}}\mathbf{b}.\_\,,\ \tau.\_\,,\ \mathbf{l_{ab}}.\_\}$**:** given a pair of processes in $\mathcal{R}$, both can do the same paths, inferred through an axiom. Their continuations are just the unprefixed processes, which are bisimilar by definition of $\mathcal{R}$.

Case op $= \mathbf{a}(\mathbf{s}).\_$**:** given $(a(s).p, a(s).q) \in \mathcal{R}$, both can do the same singleton free or bound input paths. The continuations are p and q, possibly renamed, which are again paired in $\mathcal{R}$, by Lemma A.7.

Case op $= \_ + \mathbf{q}$**:** consider $(p_1 + q, p_2 + q) \in \mathcal{R}$ and suppose $p_1 + q \stackrel{\Lambda}{\Longrightarrow} p_1'$ is inferred using (SUM-L) from $p_1 \stackrel{\Lambda}{\Longrightarrow} p_1'$. Then, by definition of $\mathcal{R}$, $p_2 \stackrel{\Lambda}{\Longrightarrow} p_2'$, with $p_1' \sim_\kappa^{NC} p_2'$, so $(p_1', p_2') \in \mathcal{R}$. The (SUM-R) case is obvious. The cases of non-linear rules, as usual, are treated by using Lemma A.2 and Lemma A.7. The case $q + \_$ is analogous.

Case op $= \_ \mid \mathbf{q}$**:** here we actually take the union of all such relations for all possible processes q. Consider $(p_1 \mid q, p_2 \mid q) \in \mathcal{R}$ and suppose $p_1 \mid q \stackrel{\Lambda_1 \mid \Lambda_2}{\Longrightarrow} p_1' \mid q'$, with $\mathrm{Bn}(\Lambda_1 \mid \Lambda_2) \# p_2 \mid q$, is inferred from $p_1 \stackrel{\Lambda_1}{\Longrightarrow} p_1'$ and $q \stackrel{\Lambda_2}{\Longrightarrow} q'$ using (PAR). Since $p_1 \sim_\kappa^{NC} p_2$ and $\mathrm{Bn}(\Lambda_1) \# p_2$, we also have $p_2 \stackrel{\Lambda_1}{\Longrightarrow} p_2'$, so we can apply (PAR) and get $p_2 \mid q \stackrel{\Lambda_1 \mid \Lambda_2}{\Longrightarrow} p_2' \mid q'$. Finally, from $p_2' \sim_\kappa^{NC} p_1'$ it follows that $(p_1' \mid q', p_2' \mid q') \in \mathcal{R}$. If the transition is inferred through non-linear rules, the usual argument applies. The case $q \mid \_$ is analogous.

Case op $= (\mathbf{r})\_$**:** directly by Lemma A.7. □

References

[1] A.T. Campbell, I. Katzela, Open signaling for atm, internet and mobile networks (opensig'98), SIGCOMM Comput. Commun. Rev. 29 (1999) 97–108.
[2] D.L. Tennenhouse, D.J. Wetherall, Towards an active network architecture, Comput. Commun. Rev. 26 (1996) 5–18.
[3] N. McKeown, T. Anderson, H. Balakrishnan, G.M. Parulkar, L.L. Peterson, J. Rexford, S. Shenker, J.S. Turner, Openflow: enabling innovation in campus networks, Comput. Commun. Rev. 38 (2) (2008) 69–74.
[4] Openflow foundation website, http://www.openflow.org/.
[5] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, i/ii, Inform. and Comput. 100 (1) (1992) 1–77.
[6] R. Milner, A Calculus of Communicating Systems, Lecture Notes in Computer Science, vol. 92, Springer, 1980.
[7] A. Francalanza, M. Hennessy, A theory of system behaviour in the presence of node and link failure, Inform. and Comput. 206 (6) (2008) 711–759.
[8] R. De Nicola, D. Gorla, R. Pugliese, Basic observables for a calculus for global computing, Inform. and Comput. 205 (10) (2007) 1491–1525.
[9] A. Corradini, U. Montanari, F. Rossi, An abstract machine for concurrent modular systems: Charm, Theoret. Comput. Sci. 122 (1&2) (1994) 165–200.
[10] M.P. Fiore, D. Turi, Semantics of name and value passing, in: LICS, IEEE Computer Society, 2001, pp. 93–104.
[11] J.J.M.M. Rutten, Universal coalgebra: a theory of systems, Theoret. Comput. Sci. 249 (1) (2000) 3–80.
[12] V. Ciancia, A. Kurz, U. Montanari, Families of symmetries as efficient models of resource binding, Electron. Notes Theor. Comput. Sci. 264 (2) (2010) 63–81.
[13] U. Montanari, M. Pistore, Structured coalgebras and minimal hd-automata for the pi-calculus, Theoret. Comput. Sci. 340 (3) (2005) 539–576.
[14] G.L. Ferrari, U. Montanari, E. Tuosto, Coalgebraic minimization of hd-automata for the pi-calculus using polymorphic types, Theoret. Comput. Sci. 331 (2–3) (2005) 325–365.
[15] I. Lanese, Synchronization strategies for global computing models, Ph.D. thesis, Computer Science Department, University of Pisa, Pisa, Italy, 2006.
[16] I. Lanese, Concurrent and located synchronizations in i-calculus, in: J. van Leeuwen, G.F. Italiano, W. van der Hoek, C. Meinel, H. Sack, F. Plasil (Eds.), SOFSEM (1), in: Lecture Notes in Computer Science, vol. 4362, Springer, 2007, pp. 388–399.
[17] U. Montanari, M. Sammartino, Network conscious π-calculus: A concurrent semantics, Electron. Notes Theor. Comput. Sci. 286 (2012) 291–306.
[18] M. Sammartino, A network-aware process calculus for global computing and its categorical framework, Ph.D. thesis, University of Pisa, 2013.
[19] S.M. Lane, Categories for the Working Mathematician, Springer-Verlag, New York, 1998.
[20] J. Adámek, Introduction to coalgebra, Theory Appl. Categ. 14 (8) (2005) 157–199.
[21] S. Staton, Relating coalgebraic notions of bisimulation, Log. Methods Comput. Sci. 7 (1) (2011) 1–21.
[22] J. Adámek, J. Rosický, Locally Presentable and Accessible Categories, Cambridge University Press, 1994.
[23] J. Worrell, Terminal sequences for accessible endofunctors, Electron. Notes Theor. Comput. Sci. 19 (1999) 24–38.
[24] S. Staton, Name-passing process calculi: operational models and structural operational semantics, Tech. Rep. 688, University of Cambridge, 2007.
[25] F. Bonchi, U. Montanari, Reactive systems, (semi-)saturated semantics and coalgebras on presheaves, Theoret. Comput. Sci. 410 (41) (2009) 4044–4066.
[26] F. Bonchi, M.G. Buscemi, V. Ciancia, F. Gadducci, A presheaf environment for the explicit fusion calculus, J. Automat. Reason. 49 (2) (2012) 161–183.
[27] F. Gadducci, M. Miculan, U. Montanari, About permutation algebras, (pre)sheaves and named sets, High.-Order Symb. Comput. 19 (2–3) (2006) 283–304.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009

[28] P. Johnstone, Sketches of an Elephant: A Topos Theory Compendium, Oxford Logic Guides, Clarendon Press, 2002.
[29] F. Bonchi, B. König, U. Montanari, Saturated semantics for reactive systems, in: LICS, 2006, pp. 69–80.
[30] Y. Rekhter, A border gateway protocol 4 (bgp-4), http://www.ietf.org/rfc/rfc1771.txt, March 1995.
[31] M. Hennessy, J. Riely, Resource access control in systems of mobile agents, Inform. and Comput. 173 (1) (2002) 82–120.
[32] R. De Nicola, G.L. Ferrari, R. Pugliese, Klaim: A kernel language for agents interaction and mobility, IEEE Trans. Software Eng. 24 (5) (1998) 315–330.
[33] R. De Nicola, D. Gorla, R. Pugliese, On the expressive power of klaim-based calculi, Theoret. Comput. Sci. 356 (3) (2006) 387–421.
[34] M. Hennessy, A calculus for costed computations, Log. Methods Comput. Sci. 7 (1) (2011) 1–35.
[35] M. Hennessy, M. Gaur, Counting the cost in the picalculus (extended abstract), Electron. Notes Theor. Comput. Sci. 229 (3) (2009) 117–129.
[36] E. De Vries, A. Francalanza, M. Hennessy, Reasoning about explicit resource management (extended abstract), in: PLACES, ETAPS, 2011, pp. 15–21.
[37] N. Ghani, K. Yemane, B. Victor, Relationally staged computations in calculi of mobile processes, Electron. Notes Theor. Comput. Sci. 106 (2004) 105–120.
[38] M. Miculan, A categorical model of the fusion calculus, Electron. Notes Theor. Comput. Sci. 218 (2008) 275–293.
[39] U. Montanari, M. Pistore, Concurrent semantics for the pi-calculus, Electron. Notes Theor. Comput. Sci. 1 (1995) 411–429.
[40] S. MacLane, I. Moerdijk, Sheaves in Geometry and Logic: A First Introduction to Topos Theory, Universitext, Springer, 1992.

Please cite this article in press as: U. Montanari, M. Sammartino, A network-conscious π-calculus and its coalgebraic semantics, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.03.009