A formal approach to open multiparty interactions

A Formal Approach to Open Multiparty Interactions ☆

Chiara Bodei[a], Linda Brodo[b], Roberto Bruni[a,*]

[a] *Dipartimento di Informatica, Università di Pisa, Italy*
[b] *Dipartimento Pol.Com.Ing., Università di Sassari, Italy*

Abstract

We present a process algebra aimed at describing interactions that are *multi-party*, i.e. that may involve more than two processes and that are *open*, i.e. the number of the processes they involve is not fixed or known a priori. Here we focus on the theory of a core version of a process calculus, without message passing, called Core Network Algebra (CNA). In CNA communication actions are given not in terms of channels but in terms of chains of links that record the source and the target ends of each hop of interactions. The operational semantics of our calculus mildly extends the one of CCS. The abstract semantics is given in the style of bisimulation but requires some ingenuity. Remarkably, the abstract semantics is a congruence for all operators of CNA and also with respect to substitutions, which is not the case for strong bisimilarity in CCS. As a motivating and running example, we illustrate the model of a simple software defined network infrastructure.

Keywords: CCS, CNA, open interaction, multi-party interaction

1. Introduction

An *interaction* is a way in which communicating processes can influence one another. Interactions in the time of the World Wide Web and of the Internet of

☆ Research partially supported by the Italian MIUR Project CINA (PRIN 2010LHT4KM), and by Università di Pisa PRA_2016_64 Project *Through the fog.*

* Corresponding Author
Email addresses: chiara@di.unipi.it (Chiara Bodei), brodo@uniss.it (Linda Brodo), bruni@di.unipi.it (Roberto Bruni)

Things (IoT) are something more than input and output between two entities. Actually, the word itself can be misleading, by suggesting a reciprocal or mutual kind of actions. Instead, interactions more and more often involve many parties, and actions are difficult to classify under output and input primitives. This is a common situation when, e.g. a client interacts with a website that in turn invokes some services from other websites. At a certain level of abstraction it is important to know which are the involved services, while it is not important how they are contacted. This practice follows the "separation of concern" modelling style, where the modeller is not interested in the details of "how" (with how many synchronisations, for example) the interaction takes place as long as a specific phase of the overall procedure is concluded with success. Intuitively, we can imagine an interaction as the composition of a jigsaw puzzle: all partners provide different pieces that fit together to complete the picture.

Networks have become part of the critical infrastructure of our daily activities (for business, home, social, health, government, etc.) and a large variety of loosely coupled processes have been offered over global networks, as services. As a consequence, more sophisticated forms of interactions have become common, for which convenient formal abstractions are under investigation. In this regard, one important trend in networking is moving towards architectures where the infrastructure itself can be manipulated by the software, as in the Software Defined Networking (SDN) approach [1]. Software clients can remotely access and modify the control plane, by using standard open protocols such as OpenFlow.[1] In this case, it is therefore possible to decouple the network control from dataflow and from the network topology and to provide Infrastructure as a Service (IaaS) over data-centers, cloud systems and IoT.

Another example, coming from a completely different research field, is that of complex biological interactions as the ones emerging in bio-computing and membrane systems, where interactions typically involve several compounds and catalysts.

[1] See, e.g. the Open Networking Foundation website `http://www.opennetworking.org`.

(a) A link (b) Matching tetrominos

Figure 1: Links as tetrominos

As a consequence, from a foundational point of view, it is strategic to provide the convenient formal abstractions and models to naturally capture these new communication patterns, by going beyond the ordinary binary form of communication, here called dyadic. These models should be sufficiently expressive to faithfully describe the complex phenomena, but they have also to provide a basis for the formal analysis of such systems, by offering sufficient mathematical structure and suitable abstraction mechanisms for tractability.

We present here a process algebra, called CNA, which takes interaction as its basic ingredient. The described interactions are *multiparty*, i.e. they may involve more than two processes and are *open*, i.e. the number of the processes they involve is not fixed or known a priori. This is not to be confused with multiparty interactions represented as a global choreography [2, 3], whose realisation is still based on dyadic interactions. Traditionally in process algebras, communication is based on synchronisation send/receive on specific channels. In CNA, instead, communication actions are given not in terms of channels but in terms of links that record the source and the target ends of each hop of interactions. Links can be indeed combined in link chains in order to describe how information can be routed across processes before arriving at destination. Note that links can be combined if they are to some extent "complementary", i.e. if each process contributes with links that are compatible, if not necessary, with the chain of links provided by the other processes. According to the puzzle analogy, different parts of a chain can be composed separately, and, afterwards, assembled by superposition without overlays.

To help the intuition, we can see links as Z-shaped tetrominos that can be

joined together along a line when the labels of the edges match (see Figure 1).

Despite the inherent complexity of representing more sophisticated forms of interaction, we show that the underlying synchronisation algebra and name handling primitives are quite simple, being a straight generalisation of dyadic ones. This is witnessed by the operational semantic rules of our calculus that, in their simpler version (i.e. without message passing), resemble the rules of CCS [4], while in the full one, not considered here (see [5]), they resemble the ones of π-calculus [6]. In this sense, CNA processes can be seen as running over a dedicated *middleware* that guarantees a proper handling of links, in the same way as CCS and π-calculus processes can be seen as running over a middleware that guarantees a proper handling of point-to-point messaging (e.g. messages are not lost).

Finally, we address a more technical issue, by providing a convenient abstract semantics, called *network bisimilarity* for CNA processes, which is the analogous of strong bisimilarity for CCS processes. Remarkably, network bisimilarity is a congruence w.r.t. all useful composition operators and also w.r.t. substitutions, a feature mostly missed in other frameworks.

Synopsis. In Section 2, we recall the basics of CCS, although we assume the reader has some familiarity with process algebras. Furthermore, we illustrate a simple scenario of a modular network infrastructure that will serve as a running example to demonstrate that the level of abstraction provided by CNA is much more convenient w.r.t. the one provided by processes with dyadic interactions.

In Section 3, we present the theory of link chains, to be used as labels in the operational semantics of CNA. The theory is quite rich, as it consists of several key operators for building and manipulating link chains. Some nice properties of the introduced operators are also proved, which later will turn out useful to assess semantics properties of CNA processes.

In Section 4, we introduce the syntax and operational semantics of CNA, together with some simple usage examples that should help the reader in understanding the driving principles behind our design choices. The key technical

contribution in this section is the Accordion Lemma 25.

In Section 5, we close the loop by introducing network bisimilarity, the abstract semantics of CNA processes. We prove the main congruence results (see Theorem 42 and Proposition 47) and show how network bisimilarity fits well in the running example. Some variations are discussed by the end of Section 5.

Discussion of related work and some concluding remarks are in Section 6.

Some auxiliary results and the proofs of technical lemmata can be found in Appendix A.

Previous work. This article is the full version of the extended abstract in [5], where also the message passing version of CNA was presented, called *link-calculus*. Here we focus on the core version of the framework and spell out its theory in full detail. It is worth mentioning that we have revised the definition of the equivalence $\rhd\lhd$ on link chains that is the basis of network bisimilarity and introduced a finer equivalence $\blacktriangleright\blacktriangleleft$ that considerably simplifies the proofs of the main properties. The motivating and running example is completely original to this contribution. Finally, we give here all proofs at a good level of detail.

The main contribution in [5] was to show that the link-calculus can be used to encode Mobile Ambients (MA) [7] in such a way that there is a bijective correspondence between the reduction steps of MA processes and silent transitions of their encodings in the link-calculus. This was a much stronger operational correspondence than any available in the literature, such as the one in [8].

In [9], following a similar line, we have provided an encoding of Brane Calculi [10] in the *link-calculus*. In particular, we have shown that biologically interactions that usually involve several compounds can be naturally rendered by multiparty interactions. Furthermore, locality can be easily handled, without introducing any specific operator, just encoding any membrane compartment as a separate process.

2. Background on CCS and a Running Example

2.1. CCS: the Calculus of Communicating Systems

The Calculus of Communicating Systems (CCS) [4] was introduced by Turing Award winner Robin Milner in the early 1980s. It is based on the notion of processes that communicate on shared channels by executing actions and co-actions over them. Without loss of generality, we can imagine them as input and output actions with synchronous dyadic interaction. Let $\mathcal{C} = \{a, b, ...\}$ be the set of channels and, by coercion, input actions. We denote by $\overline{\mathcal{C}} = \{\overline{a}, \overline{b}, ...\}$ the set of co-actions (i.e. output actions), with $\mathcal{C} \cap \overline{\mathcal{C}} = \emptyset$ and let $\mathcal{O} = \mathcal{C} \cup \overline{\mathcal{C}}$ denote the set of observable actions ranged over by λ. We extend the bar-notation to observable actions, by letting $\overline{\overline{\lambda}} = \lambda$. We also fix a distinguished silent action $\tau \notin \mathcal{O}$ and let $\mu \in \mathcal{O} \cup \{\tau\}$ denote a generic action. A *channel relabelling* is a function $\phi : \mathcal{C} \to \mathcal{C}$. It is extended to generic actions by letting $\phi(\tau) = \tau$ and $\phi(\overline{\lambda}) = \overline{\phi(\lambda)}$ for any observable action λ. It is called a *renaming* when it is bijective.

A CCS process is then a term generated by the following grammar:

$$p, q \quad ::= \quad \mathbf{0} \mid \mu.p \mid p + q \mid p|q \mid (\nu\, a)p \mid p[\phi] \mid A$$

where ϕ is a channel relabelling function and A is any constant drawn from a set Δ of possibly recursive definitions of the form $A \triangleq p$.

Roughly the process $\mathbf{0}$ is the inactive process that cannot perform any action. The action prefixed process $\mu.p$ can execute the action μ and then behaves as p. The operator $+$ introduces nondeterminism: the composed process $p + q$ can behave as p or as q, but once it performs an action as p the option q is discarded, and vice versa. The parallel composition of two processes, written $p|q$, allows p and q to interleave their actions or to interact by performing complementary actions a and $\overline{a}$: if this is the case, the synchronisation is represented as a silent action τ and the channel where it takes places is not recorded. The restricted process $(\nu\, a)p$ can perform all actions that p can perform, except for actions a and $\overline{a}$, which are blocked. The relabelled process $p[\phi]$ can perform all actions

$$\frac{}{\mu.p \xrightarrow{\mu} p} \qquad \frac{p \xrightarrow{\mu} p'}{p+q \xrightarrow{\mu} p'} \qquad \frac{q \xrightarrow{\mu} q'}{p+q \xrightarrow{\mu} q'} \qquad \frac{p \xrightarrow{\mu} p' \qquad \mu \notin \{a, \overline{a}\}}{(\nu\, a)p \xrightarrow{\mu} (\nu\, a)p'}$$

$$\frac{p \xrightarrow{\mu} p'}{p[\phi] \xrightarrow{\phi(\mu)} p'[\phi]} \qquad \frac{p \xrightarrow{\mu} p'}{p|q \xrightarrow{\mu} p'|q} \qquad \frac{q \xrightarrow{\mu} q'}{p|q \xrightarrow{\mu} p|q'} \qquad \frac{p \xrightarrow{\lambda} p' \qquad q \xrightarrow{\overline{\lambda}} q'}{p|q \xrightarrow{\tau} p'|q'}$$

$$\frac{p \xrightarrow{\mu} q \qquad (A \triangleq p) \in \Delta}{A \xrightarrow{\mu} q}$$

Figure 2: SOS semantics of CCS.

that p can perform, but they are relabelled according to ϕ, i.e. if p can do an action μ then $p[\phi]$ can do $\phi(\mu)$. Relabelling is very useful for reusing process components in different parts of the system just by changing the set of channels on which they operate. Finally, the constant A behaves as p if $(A \triangleq p) \in \Delta$.

In some cases we shall use constants $A(x_1, ..., x_n)$ that are parametric on a set of channel names $x_1, ..., x_n$, written more concisely as $A(\widetilde{x})$, and that can be instantiated with actual names, as in $A(a_1, ..., a_n)$ or just $A(\widetilde{a})$ for short. Similarly, we write $(\nu\, \widetilde{a})p$ for $(\nu\, a_1) \cdots (\nu\, a_n)p$.

The operational semantics of CCS is given in the form of a Labelled Transition System (LTS), where the states are CCS processes and the transitions are labelled by actions. We write $p \xrightarrow{\mu} q$ if p can perform the action μ and behave as q afterwards. The inference rules that generate the LTS are defined in the style of Structural Operational Semantics (SOS) as they are driven by the syntax of processes (see Figure 2).

For example, we have transitions such as $a.b.\mathbf{0} \xrightarrow{a} b.\mathbf{0} \xrightarrow{b} \mathbf{0}$, $a.b.\mathbf{0} + c.\mathbf{0} \xrightarrow{a} b.\mathbf{0}$ and $a.b.\mathbf{0} + c.\mathbf{0} \xrightarrow{c} \mathbf{0}$. The interplay between restriction and parallel composition is interesting as it can be used to impose synchronisation on some channel. In fact, while for $A \triangleq (a.b.\mathbf{0} + c.\mathbf{0})|(\overline{a}.\mathbf{0} + d.\mathbf{0})$ we have transitions such as $A \xrightarrow{a} b.\mathbf{0}|(\overline{a}.\mathbf{0} + d.\mathbf{0})$, $A \xrightarrow{\overline{a}} (a.b.\mathbf{0} + c.\mathbf{0})|\mathbf{0}$, and $A \xrightarrow{\tau} b.\mathbf{0}|\mathbf{0}$, among others, the process $(\nu\, a)A$ cannot perform any action labelled by a and $\overline{a}$ but can still perform the synchronisation $(\nu\, a)A \xrightarrow{\tau} (\nu\, a)(b.\mathbf{0}|\mathbf{0})$, because τ actions cannot be

restricted.

To keep the notation compact, we write $p \xrightarrow{\mu_1} \xrightarrow{\mu_2} \cdots \xrightarrow{\mu_n} q$ when there exist some processes $p_1, ...p_{n+1}$ that we do not need to mention such that $p_1 = p$, $p_{n+1} = q$ and $p_i \xrightarrow{\mu_i} p_{i+1}$ for $i \in [1, n]$.

Recursive definitions can be used to account for infinite behaviour. For example, if $(A \triangleq a.A + b.\mathbf{0}) \in \Delta$, then the process A can do any finite sequence of actions a terminated by an action b, as in $A \xrightarrow{a} A \xrightarrow{a} \cdots \xrightarrow{a} A \xrightarrow{b} \mathbf{0}$ but it can also perform an infinite sequence of actions a, as in $A \xrightarrow{a} A \xrightarrow{a} \cdots \xrightarrow{a} A \xrightarrow{a} \cdots$.

The main notion of equivalence for CCS processes is called *strong bisimilarity* and is denoted by $\sim$. It is defined as the largest strong bisimulation relation, i.e. as the largest binary relation $\mathbf{R}$ on CCS processes such that whenever $p \,\mathbf{R}\, q$ we have that:

1. for any μ, p' such that $p \xrightarrow{\mu} p'$ there exists q' such that $q \xrightarrow{\mu} q'$ and $p' \,\mathbf{R}\, q'$;
2. for any μ, q' such that $q \xrightarrow{\mu} q'$ there exists p' such that $p \xrightarrow{\mu} p'$ and $p' \,\mathbf{R}\, q'$.

Notably, strong bisimilarity is a congruence w.r.t. all the operators of CCS. Here we point out that it is not a congruence w.r.t. action substitution. For example, it is well-known that strong bisimilarity reduces concurrency to non-determinism, as $a.\mathbf{0} \mid \overline{b}.\mathbf{0} \sim a.\overline{b}.\mathbf{0} + \overline{b}.a.\mathbf{0}$. However, if we apply the (non-injective) substitution $\{b/a\}$ that replaces all the (free) occurrences of a with b to both processes we get $b.\mathbf{0} \mid \overline{b}.\mathbf{0} \not\sim b.\overline{b}.\mathbf{0} + \overline{b}.b.\mathbf{0}$, because the former process can do the silent step $b.\mathbf{0} \mid \overline{b}.\mathbf{0} \xrightarrow{\tau} \mathbf{0} \mid \mathbf{0}$, while the latter process $b.\overline{b}.\mathbf{0} + \overline{b}.b.\mathbf{0}$ cannot.

Sometimes one wants to abstract away from silent transitions τ. Correspondingly *weak bisimilarity* can be considered instead of strong bisimilarity, where in the bisimulation game a single transition can be simulated by exploiting any number of silent transitions. Unfortunately, weak bisimilarity is not a congruence w.r.t. the choice operator and substitutions.

2.2. Software Defined Infrastructures

In this sub-section we sketch four scenarios of increasing complexity together with their possible modelling in CCS. Once introduced CNA, in Sections 4 and 5,

we will revisit these examples to show that they can be more conveniently accounted for in CNA.

The reference case study consists of a network infrastructure with n requestor agents $A_1, ..., A_n$, m servers $S_1, ..., S_m$ and a routing infrastructure R that regulates which requestors are connected to which servers in a way that is out of the control of agents and requestors. For the sake of simplicity, in the following we let $n = m = 2$.

Example 1 (Blind routing). Initially, we keep the scenario as simple as possible: the idea is that a requestor can repeatedly request a service if there is a non-busy server connected to it via R. Let us suppose that R connects A_1 with S_1 and S_2, while A_2 only with S_2. In CCS, the system can be readily modelled by the following recursive processes that run in parallel.

$$\begin{aligned} A_i &\triangleq \overline{req_i}.\overline{think}.A_i \quad \text{for } i \in [1,2] \\ S_j &\triangleq srv_j.\overline{exec}.S_j + \overline{busy}.\tau.S_j \quad \text{for } j \in [1,2] \\ R &\triangleq req_1.(\overline{srv_1}.R + \overline{srv_2}.R) + req_2.\overline{srv_2}.R \end{aligned}$$

For example we can let the system be defined as

$$N \triangleq (\nu\, \widetilde{req})(\nu\, \widetilde{srv})(A_1 \mid A_2 \mid R \mid S_1 \mid S_2)$$

so that synchronisation is enforced for all the interactions between requestors and the infrastructure (on channels req_1 and req_2) and between the infrastructure and servers (on channels srv_1 and srv_2).

The routing depends on the state of the system. Suppose, for instance, that S_2 becomes busy and that A_2 sends a request to R, according to the transition sequence $N \xrightarrow{\overline{busy}} \xrightarrow{\tau} N'$ with

$$N' = (\nu\, \widetilde{req})(\nu\, \widetilde{srv})(A_1 \mid \overline{think}.A_2 \mid \overline{srv_2}.R \mid S_1 \mid \tau.S_2).$$

This is perfectly admissible, but leaves A_2 thinking its request has been served, because the interaction with R has taken place, while the server S_2 has not even received it.

Example 2 (Acknowledged routing). To remedy the problem raised by the previous model, one can introduce some acknowledgement protocol, to ensure each agent that its request has been assigned to some server. The CCS model can thus be improved by redesigning the processes as follows:

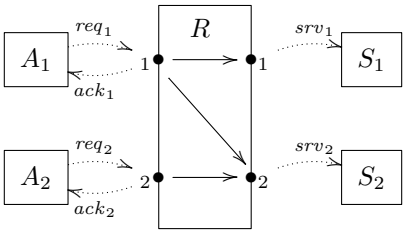
$$\begin{aligned} A_i &\triangleq \overline{req_i}.ack_i.\overline{think}.A_i \quad \text{for } i \in [1,2] \\ S_j &\triangleq srv_j.\overline{exec}.S_j + \overline{busy}.\tau.S_j \quad \text{for } j \in [1,2] \\ R &\triangleq req_1.(\overline{srv_1}.\overline{ack_1}.R + \overline{srv_2}.\overline{ack_1}.R) + req_2.\overline{srv_2}.\overline{ack_2}.R \end{aligned}$$

For example we can let the system be defined as

$$M \triangleq (\nu\, \widetilde{ack})N$$

where N is defined as before.

At a very abstract level, we can view an infrastructure as an oriented graph with n nodes on the left boundary and m nodes on the right boundary: the assignment of a request from A_i to the server S_j is possible if S_j is available and if there is a connection between the ith node on the left boundary and the jth node on the right boundary. Graphically, M can be depicted as below:



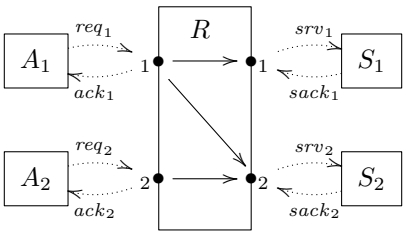
This time, when S_2 is busy and A_2 interacts with the infrastructure R on channel req_2, the requestor A_2 blocks until the server S_2 becomes available and can accept the request by interacting on channel srv_2 with R. In fact, when this is the case, R sends the acknowledgment on channel ack_2 to A_2.

Example 3 (Composite, acknowledged routing). Now suppose that the infrastructure R is not monolithic, and that it is instead obtained by composing some network infrastructures together, which is a necessity for complex systems.

To make the infrastructure compositional, we must make interaction symmetric on the two, left and right, boundaries, i.e. we must assume that servers also send some acknowledgement. Correspondingly, we set

$$S_j \quad = \quad srv_j.\overline{sack_j}.\overline{exec}.S_j + \overline{busy}.\tau.S_j \qquad \text{for } j \in [1,2]$$

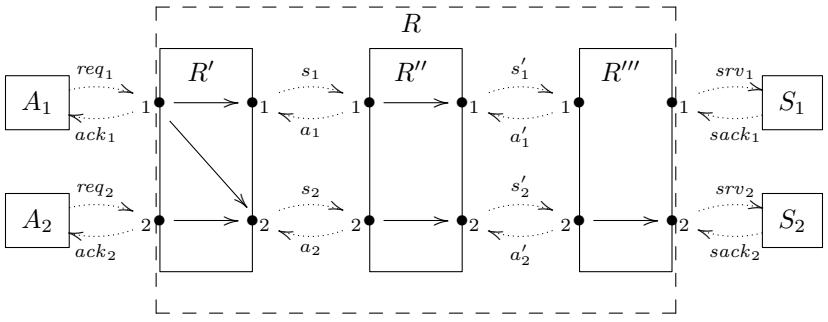
225 Now the system can be depicted as below:



Now consider the case where R is obtained by juxtaposing three other infrastructures R', R'' and R''' defined as follows:

$$\begin{aligned}
R' &\triangleq req_1.(\overline{s_1}.a_1.\overline{ack_1}.R' + \overline{s_2}.a_2.\overline{ack_1}.R') + req_2.\overline{s_2}.a_2.\overline{ack_2}.R' \\
R'' &\triangleq s_1.\overline{s'_1}.a'_1.\overline{a_1}.R'' + s_2.\overline{s'_2}.a'_2.\overline{a_2}.R' \\
R''' &\triangleq s'_2.\overline{srv_2}.sack_2.\overline{a'_2}.R''' \\
R &\triangleq (\nu\, \widetilde{a'})(\nu\, \widetilde{s'})(\nu\, \widetilde{a})(\nu\, \widetilde{s})(R' \mid R'' \mid R''')
\end{aligned}$$

Note that R''' does not forward any request coming from its first port. The resulting infrastructure is illustrated in the figure below:



In general, an assignment of a request to a server is possible only if there is a path of connections in the graph associated with the infrastructure. In the

example, the requests coming from A_1 and A_2 can only be assigned to S_2, as there is no path towards S_1.

Unfortunately, it may happen that the routing of the infrastructure comes to a dead point. In the example, R' can forward the request from A_1 to R'' on s_1, but then R'' is blocked because it will not be able to pass the request to R''' on s'_1.

To remedy this, either all dead paths must be removed before the infrastructure is deployed or some deadlock-detection and backtracking mechanism should be put in place, which requires some additional efforts.

Example 4 (Dynamic routing). Finally, imagine the situation where the infrastructure R is *software defined*, in the sense that connections can be added and removed dynamically. This time static-time dead-path analysis is not possible at all, and the integration of this additional feature with the previous acknowledgement, deadlock-detection and backtracking mechanisms looks overly complicated.

3. A Theory of Link Chains

To address the challenges posed by the scenarios in Section 2, the idea is to move from dyadic interaction to multiparty one. Correspondingly, communication actions are given in terms of *links* and a single atomic interaction is possibly composed by more than one link. A link is a pair ${}^{\alpha}\backslash_{\beta}$ that records the source and the target sites of a communication, meaning that the input available at the source site α can be forwarded to the target one β. Links are suitably combined in link chains to describe how information can be routed across processes before arriving at their destination. Therefore, links are combined like pieces in a jigsaw puzzle, where each party contributes with its link. As explained in the introduction, we can think about links as Z-shaped tetrominos that are joined together in a line when the labels of the edges match so to form a link chain. Standard I/O communication is made more accurate, by recording

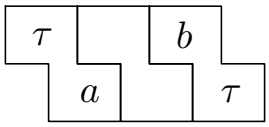


Figure 3: A chain with a missing link.

the route of information across several sites. Furthermore, link chains allow seamless realisation of multiparty synchronisations.

To achieve compositionality, we allow processes to provide link chains that are assembled just in part. Intuitively they correspond to puzzles where some, but not all, the pieces are present. As an example, Figure 3 shows a chain with a missing link from a to b.

In this section we present the underlying theory of links and link chains, posing the emphasis on the operations for combining them and on some relevant properties they satisfy.

3.1. Links

Let $\mathcal{C}$ be the set of channels, ranged over by $a, b, c, ...$, and let $\mathcal{A} = \mathcal{C} \cup \{\tau\} \cup \{\Box\}$ be the set of actions, ranged over by $\alpha, \beta, \gamma, ...$, where the symbol τ denotes a *silent* action, while the symbol $\Box$ denotes a *virtual* (non-specified) action (i.e. a missing piece of the puzzle according to the analogy proposed above).

Definition 5 (Links: solid, virtual, valid). A *link* is a pair $\ell = {}^{\alpha}\backslash_{\beta}$; it can be read as forwarding the input available on α to β, and we call α the *source site* of ℓ and β the *target site* of ℓ. A link ${}^{\alpha}\backslash_{\beta}$ is *solid* if $\alpha, \beta \neq \Box$; the link ${}^{\Box}\backslash_{\Box}$ is called *virtual*. A link is *valid* if it is solid or virtual. We let $\mathcal{L}$ be the set of valid links.

Examples of non valid links are ${}^{\tau}\backslash_{\Box}$ and ${}^{\Box}\backslash_{a}$, while both ${}^{\tau}\backslash_{a}$ and ${}^{a}\backslash_{b}$ are solid valid links. From now on, we only consider valid links.

As it will be shortly explained, the virtual link ${}^{\Box}\backslash_{\Box}$ is a sort of "missing link" inside a link chain; it represents a needed link that can be supplied, as

a solid link, by another link chain, via a suitable composition operation called *merge* (see below).

3.2. Link Chains

Links can be combined in link chains that record the source and the target sites of each hop of the interaction.

Definition 6 (Link Chain). A *link chain* is a finite sequence $s = \ell_1 ... \ell_n$ of (valid) links $\ell_i = {}^{\alpha_i}\backslash_{\beta_i}$ such that:

1. for any $i \in [1, n-1]$, $\begin{cases} \beta_i, \alpha_{i+1} \in \mathcal{C} & \text{implies } \beta_i = \alpha_{i+1} \\ \beta_i = \tau & \text{iff } \alpha_{i+1} = \tau \end{cases}$
2. $\exists i \in [1, n]$. $\ell_i \neq {}^{\Box}\backslash_{\Box}$.

The first condition says that any two adjacent solid links must match on their adjacent sites; it also imposes that, in particular, τ cannot be matched by $\Box$. The second condition disallows chains made of virtual links only. A non-empty link chain is *solid* if all its links are so. For example, ${}^{\tau}\backslash^{a}_{a}\backslash_{b}$ is a solid link chain, while ${}^{\tau}\backslash^{\Box}_{a}\backslash^{b}_{\Box}\backslash_{\tau}$ is not solid.

In counting links in a chain we may decide to ignore or not virtual links.

Definition 7 (Length and size). The *length* of a chain s, written $|s|$, is the number of valid (virtual and solid) links that are in s. The *size* of s, written $||s||$, is the number of solid links that are in s.

For example, $|{}^{\tau}\backslash^{\Box}_{a}\backslash^{b}_{\Box}\backslash_{\tau}| = 3$, while $||{}^{\tau}\backslash^{\Box}_{a}\backslash^{b}_{\Box}\backslash_{\tau}|| = 2$.

The following definition introduces an equivalence relation over link chains that equates two valid link chains if they only differ for the presence of virtual links only.

Definition 8 (Equivalence $\bowtie$). We let $\bowtie$ be the least equivalence relation over link chains closed under the axioms (whenever both sides are well defined link chains):

$$s^{\Box}\backslash_{\Box} \bowtie s \qquad s_1{}^{\Box}\backslash^{\Box}_{\Box}\backslash_{\Box} s_2 \bowtie s_1{}^{\Box}\backslash_{\Box} s_2$$

$${}^{\Box}\backslash_{\Box} s \bowtie s \qquad s_1{}^{\alpha}\backslash^{a}_{a}\backslash_{\beta} s_2 \bowtie s_1{}^{\alpha}\backslash^{\Box}_{a}\backslash^{a}_{\Box}\backslash_{\beta} s_2$$

From the above definition, it follows that the chain size is invariant w.r.t. $\bowtie$, i.e. $s \bowtie s'$ implies that $||s|| = ||s'||$ (although s and s' can have different lengths). Furthermore, for ℓ a solid link and s a link chain, we write $s \bowtie \ell$ if and only if ℓ is the *only* solid link that occurs in s.

The following basic operations over links and link chains are partial and strict, i.e. they may issue $\perp$ (undefined) and the result is $\perp$ if any argument is $\perp$. To keep the notation short, we tacitly assume that the result is $\perp$ if either one of the sub-expressions in the righthand side of any defining equation is undefined, or if none of the conditions in the righthand side of any defining equation is met.

Merge. We remind that the virtual links in a chain can be seen as the part in the chain not yet specified, and possibly provided by another link chain when merged.

Two link chains can be merged if they are to some extent "complementary", in the sense that: (i) they have the same length; (ii) each of them provides solid links that are missing in the other, and (iii) superimposed together they still form a link chain.

In particular, if there is a position where both link chains carry solid links, then there is a clash and the merge is not possible (undefined). Also if the merge would result in a non valid sequence, then the merge is not possible.

Definition 9 (Merge). For $s = \ell_1 ... \ell_n$ and $s' = \ell'_1 ... \ell'_n$, with $\ell_i = {}^{\alpha_i}\backslash_{\beta_i}$ and $\ell'_i = {}^{\alpha'_i}\backslash_{\beta'_i}$ for any $i \in [1, n]$, we define their merge $s \bullet s'$ by defining the merge of two actions as follows:

$$\alpha \bullet \beta \quad \triangleq \quad \begin{cases} \alpha & \text{if } \beta = \square \\ \beta & \text{if } \alpha = \square \end{cases}$$

and then taking its homomorphic extension to links and link chains:[2]

$$s \bullet s' \quad \triangleq \quad (\ell_1 \bullet \ell'_1) \cdots (\ell_n \bullet \ell'_n) \qquad {}^{\alpha}\backslash_{\beta} \bullet {}^{\alpha'}\backslash_{\beta'} \quad \triangleq \quad {}^{(\alpha \bullet \alpha')}\backslash_{(\beta \bullet \beta')}$$

[2] As anticipated, we remark that in the defining equations for merge it is implicitly understood that: if $\ell_i \bullet \ell'_i = \perp$ for some i, then $s \bullet s' = \perp$; if the sequence $(\ell_1 \bullet \ell'_1) ... (\ell_n \bullet \ell'_n)$ is not

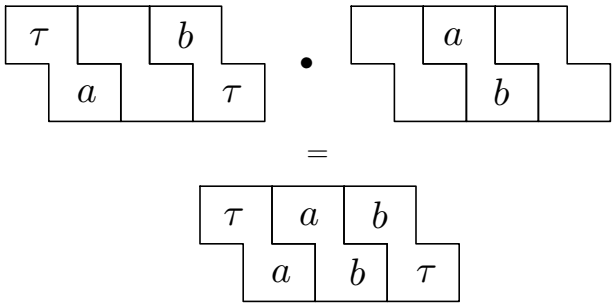


Figure 4: Merge as assembling tetrominos.

Roughly, the merge is defined element-wise on the actions of a link chain, by ensuring that whenever two actions are merged, (at least) one of them is $\square$ and that the result of the merge is still a link chain. Note that the merge is undefined if the link chains have different lengths.

Intuitively, we can imagine that s and s' are two parts of the same puzzle separately assembled, where solid links are the pieces of the puzzle and virtual links are the holes in the puzzle and their merge $s \bullet s'$ puts the two matched parts together, without piece overlaps (see Figure 4).

Example 10. Let $s_1 = {}^{\tau}\backslash^{\square}_{a}\backslash^{\square}_{\square}\backslash_{\square}$, $s_2 = {}^{\square}\backslash^{a}_{\square}\backslash^{\square}_{b}\backslash_{\square}$, and $s_3 = {}^{\square}\backslash^{\square}_{\square}\backslash^{b}_{\square}\backslash_{\tau}$ be three link chains of the same length $|s_1| = |s_2| = |s_3| = 3$. Then s_1 and s_2 can be merged to obtain $s = s_1 \bullet s_2 = ({}^{\tau}\backslash_{a} \bullet {}^{\square}\backslash_{\square})({}^{\square}\backslash_{\square} \bullet {}^{a}\backslash_{b})({}^{\square}\backslash_{\square} \bullet {}^{\square}\backslash_{\square}) = ({}^{\tau\bullet\square}\backslash_{a\bullet\square})({}^{\square\bullet a}\backslash_{\square\bullet b})({}^{\square\bullet\square}\backslash_{\square\bullet\square}) = {}^{\tau}\backslash^{a}_{a}\backslash^{\square}_{b}\backslash_{\square}$. Similarly, s and s_3 can then be merged to obtain: $s \bullet s_3 = {}^{\tau}\backslash^{a}_{a}\backslash^{b}_{b}\backslash_{\tau}$.

The merge operation enjoys some simple algebraic properties.

Lemma 11. *For any ℓ, ℓ', s, s':*

(i) The merge of links and link chains is commutative and associative.

(ii) $\ell \bullet \ell' = {}^{\square}\backslash_{\square}$ if and only if $\ell = \ell' = {}^{\square}\backslash_{\square}$.

---

a link chain, then $s \bullet s' = \bot$; if $\alpha \bullet \alpha' = \bot$ or $\beta \bullet \beta' = \bot$, then ${}^{\alpha}\backslash_{\beta} \bullet {}^{\alpha'}\backslash_{\beta'} = \bot$; if $\alpha, \beta \neq \square$, then $\alpha \bullet \beta = \bot$.

(iii) If s is solid, then for any s' we have $s \bullet s' = \bot$.

Finally, the following lemma about the composition of link chains will be exploited in Lemma 25 to prove that the operational semantics of CNA is insensitive w.r.t. the equivalence $\bowtie$.

Lemma 12. *Let s, s', and s'' be three link chains such that $(s' \bullet s'') \bowtie s$, then there must exist s_1 and s_2 such that $s_1 \bowtie s'$ and $s_2 \bowtie s''$ with $s_1 \bullet s_2 = s$.*

Restriction. Certain actions of the link chain can be hidden by restricting the channel where they take place. Of course, restriction of a is possible only if no pending communication on a (like ${}^{\tau}\backslash^{\Box}_{a}\backslash_{\Box}$) is present, i.e. only matched communication pairs, in intermediate positions, can be restricted (as in ${}^{\tau}\backslash^{a}_{a}\backslash_{\tau}$).

Definition 13 (Matched Action). Let $s = \ell_1 ... \ell_n$, with $\ell_i = {}^{\alpha_i}\backslash_{\beta_i}$ for $i \in [1, n]$. We say that an action a is *matched* in s if:

1. $a \neq \alpha_1, \beta_n$, and
2. for any $i \in [1, n-1]$, either $\beta_i = \alpha_{i+1} = a$ or $\beta_i, \alpha_{i+1} \neq a$.

Otherwise, we say that a is *unmatched* (or *pending*) in s.

It follows from the definition that we say that a is matched in s also when a does not appear at all in s.

For instance, a is matched in the sequence ${}^{\tau}\backslash^{a}_{a}\backslash_{\tau}$, while it is pending in the sequences ${}^{\tau}\backslash^{\Box}_{a}\backslash_{\Box}$ and in ${}^{a}\backslash^{a}_{a}\backslash^{a}_{a}\backslash_{a}$.

Definition 14 (Restriction). Let $s = \ell_1 ... \ell_n$, with $\ell_i = {}^{\alpha_i}\backslash_{\beta_i}$ with $i \in [1, n]$. We define the restriction operation $(\nu\, a)s$ by letting

$$(\nu\, a)s \triangleq \begin{cases} ((\nu\, a)\ell_1) \ldots ((\nu\, a)\ell_n) & \text{if } a \text{ is } \textit{matched} \text{ in } s \\ \bot & \text{otherwise} \end{cases}$$

where:

$$(\nu\, a)^{\alpha}\backslash_{\beta} \triangleq {}^{((\nu\, a)\alpha)}\backslash_{((\nu\, a)\beta)} \qquad (\nu\, a)\alpha \triangleq \begin{cases} \tau & \text{if } \alpha = a \\ \alpha & \text{otherwise} \end{cases}$$

Restriction on links enjoy properties similar to the usual structural congruence laws for processes.

Lemma 15. *For any a, b, ℓ, s, s'*

(i) $(\nu\, a)\ell = {}^{\Box}\backslash_{\Box}$ *if and only if* $\ell = {}^{\Box}\backslash_{\Box}$.

(ii) $(\nu\, a)(s \bullet s') = s \bullet (\nu\, a)s'$ *if* a *does not occur in* s.

(iii) $(\nu\, a)(\nu\, b)s = (\nu\, b)(\nu\, a)s$.

Example 16. Let $s = {}^{\tau}\backslash^{a}_{a}\backslash^{\Box}_{b}\backslash_{\Box}$ and $s' = {}^{\Box}\backslash^{\Box}_{\Box}\backslash^{b}_{\Box}\backslash_{\tau}$. Then, we have that $(\nu\, a)s = ((\nu\, a)^{\tau}\backslash_{a})((\nu\, a)^{a}\backslash_{b})((\nu\, a)^{\Box}\backslash_{\Box}) = {}^{\tau}\backslash^{\tau}_{\tau}\backslash^{\Box}_{b}\backslash_{\Box}$, while $(\nu\, a)(s \bullet s') = {}^{\tau}\backslash^{\tau}_{\tau}\backslash^{b}_{b}\backslash_{\tau} = ((\nu\, a)s) \bullet s'$, because a does not occur in s'.

Finally, we prove a technical lemma, similar to Lemma 12 for the merge, that will be exploited in the proof of Lemma 25.

Lemma 17. *Let s and s' be two link chains such that $(\nu\, a)s$ is defined and $(\nu\, a)s \blacktriangleright\!\blacktriangleleft s'$, then there exists s'' such that $s' = (\nu\, a)s''$ and $s \blacktriangleright\!\blacktriangleleft s''$.*

Renaming. The last operation that we present is called *renaming* and allows us to change, in a uniform way, the channel names appearing in a link chain. A *channel renaming function* is a bijection $\phi : \mathcal{A} \to \mathcal{A}$ such that $\phi(\tau) = \tau$ and $\phi(\Box) = \Box$.

Definition 18 (Renaming). Let $s = \ell_1 ... \ell_n$, with $\ell_i = {}^{\alpha_i}\backslash_{\beta_i}$ with $i \in [1, n]$, and ϕ be a channel renaming function. We define the renaming operation $s[\phi]$ by letting

$$s[\phi] \triangleq (\ell_1[\phi]) \ldots (\ell_n[\phi]) \qquad ({}^{\alpha}\backslash_{\beta})[\phi] \triangleq {}^{\phi(\alpha)}\backslash_{\phi(\beta)}$$

It can be readily checked that channel renaming functions enjoy the following properties.

Lemma 19. *For any $a, \phi, \psi, \ell, s, s'$*

(i) $\ell[\phi] = {}^{\Box}\backslash_{\Box}$ *if and only if* $\ell = {}^{\Box}\backslash_{\Box}$.

(ii) $(s \bullet s')[\phi] = s[\phi] \bullet (s'[\phi])$.

(iii) $((\nu\, a)s)[\phi] = (\nu\, \phi(a))(s[\phi])$.

(iv) $s[\phi][\psi] = s[\psi \circ \phi]$.

(v) *If* $s \bowtie s'$ *then* $s[\phi] \bowtie s'[\phi]$.

We conclude this section by proving a last technical lemma that will be exploited in the proof of Lemma 25.

Lemma 20. *Let ϕ be a channel renaming function and s, s' be two link chains such that $s[\phi] \bowtie s'$, then there exists s'' such that $s' = s''[\phi]$ and $s \bowtie s''$.*

4. A Core Network Algebra

In this section we build on the theory of links and link chains to present the syntax and operational semantics of CNA.

4.1. Syntax

Definition 21. The CNA processes are generated by the following grammar:

$$P, Q \quad ::= \quad \mathbf{0} \mid \ell.P \mid P + Q \mid P|Q \mid (\nu\, a)P \mid P[\phi] \mid A$$

where ℓ is a solid link (i.e. $\ell = {}^{\alpha}\backslash_{\beta}$ with $\alpha, \beta \neq \Box$), ϕ is a channel renaming function, and A is a process identifier for which we assume a definition $A \triangleq P$ is available in a given set Δ of (possibly recursive) process definitions.

As usual, we write $\tilde{a}$ for tuples of channels and we allow parametric process definitions of the form $A(\tilde{a}) \triangleq P$, where $\tilde{a}$ is the set of free channels of P. For brevity, in the examples, we sometimes write $A \triangleq P$ leaving implicit that the free channles of P are the parameters of A.

Remark 22. The extension in which generic link chains are allowed as action prefixes instead of solid links is discussed in Section 5.3.

It is evident that processes are built over a CCS-like syntax, with inactive process $\mathbf{0}$, action prefix $\ell.P$, choice $P + Q$, parallel $P|Q$, restriction $(\nu\, a)P$,

renaming $P[\phi]$ and constant definition A, but where the underlying synchronisation algebra [11] is based on link chains. This is made evident by the operational semantics that we present next.

As usual, $(\nu\, a)P$ binds the occurrences of a in P, the sets of free and of bound names of a process P are defined in the obvious way and processes are taken up to alpha-conversion of bound names. We shall sometimes omit trailing $\mathbf{0}$, e.g. by writing ${}^{a}\backslash_{b}$ instead of ${}^{a}\backslash_{b}.\mathbf{0}$.

Example 23. CNA provides us with a natural way to rephrase the communication primitives of usual process calculi, such as CCS and CSP [12], in terms of links.

- Intuitively, the output action $\overline{a}$ (resp. the input action a) of CCS can be seen as the link ${}^{\tau}\backslash_{a}$ (resp. ${}^{a}\backslash_{\tau}$) and the solid link chain ${}^{\tau}\backslash^{a}_{a}\backslash_{\tau}$ as a dyadic communication, analogous to the silent action τ of CCS.

- The action a of CSP can be seen as the link ${}^{a}\backslash_{a}$ and the solid link chain ${}^{a}\backslash^{a}_{a}\backslash^{a}_{a}\backslash_{a}$ as a CSP-like communication among three peers over a.

4.2. Operational Semantics

The idea is that communication can be routed across several processes by combining the links they make available to form a link chain. Since the length of the link chain is not fixed *a priori*, an open multi-party synchronisation is realised.

The operational semantics is defined in terms of a Labelled Transition System, in which states are CNA processes, labels are link chains, and transitions are generated by the SOS rules in Figure 5. Notice that the rules are very similar to the ones of CCS, apart from the labels that record the link chains involved in the transitions: moving from dyadic to *linked* interaction does not introduce any complexity burden from the formal point of view.

We comment in details the rules (Act), (Res), and (Com). In rules (Res) and (Com) we leave implicit the side conditions $(\nu\, a)s \neq \bot$ and $s \bullet s' \neq \bot$,

$$\frac{s \bowtie \ell}{\ell.P \xrightarrow{s} P}\ (Act) \qquad \frac{P \xrightarrow{s} P'}{P + Q \xrightarrow{s} P'}\ (Lsum) \qquad \frac{P \xrightarrow{s} P'}{(\nu\, a)P \xrightarrow{(\nu\, a)s} (\nu\, a)P'}\ (Res)$$

$$\frac{P \xrightarrow{s} P'}{P[\phi] \xrightarrow{s[\phi]} P'[\phi]}\ (Ren) \qquad \frac{P \xrightarrow{s} P'}{P|Q \xrightarrow{s} P'|Q}\ (Lpar) \qquad \frac{P \xrightarrow{s} P' \qquad Q \xrightarrow{s'} Q'}{P|Q \xrightarrow{s \bullet s'} P'|Q'}\ (Com)$$

$$\frac{P \xrightarrow{s} P' \qquad (A \triangleq P) \in \Delta}{A \xrightarrow{s} P'}\ (Ide)$$

Figure 5: SOS semantics of the CNA (rules (*Rsum*) and (*Rpar*) are omitted).

respectively (they can be easily recovered by noting that otherwise the label of the transition in the conclusion would be undefined).

The rule (*Act*) states that $\ell.P \xrightarrow{s} P$ for any link chain s, whose unique solid link is ℓ, i.e. any s such that $s \bowtie \ell$ (we recall that $s \bowtie \ell$ if s and ℓ differ only for the presence of virtual links). Intuitively, $\ell.P$ can take part in any interaction, in any (admissible) position. To join in a communication, $\ell.P$ should exhibit the capability to enlarge its link ℓ to a link chain $s \bowtie \ell$, whose length is the same as the length of the chains offered by all the other participants, so to proceed with the merge operation. Following the early style, the suitable length is inferred at the time of deducing the input transition. Note that, by definition of link chain, if one site of ℓ is τ, then ℓ can only appear at one of the extremes of s.

The rule (*Res*) can serve different aims: (i) *floating*, if a does not occur in s, then $(\nu\, a)s = s$ and $(\nu\, a)P \xrightarrow{s} (\nu\, a)P'$; (ii) *hiding*, if a is matched in s (i.e. a appears as sites already matched by adjacent links), then all occurrences of a in s are transformed to τ in $(\nu\, a)s$; (iii) *blocking*, if a is pending in s (i.e. there are some unmatched occurrences of a in s), then $(\nu\, a)s = \bot$ and the rule cannot be applied.

In the (*Com*) rule the link chains recorded on both the premises' transitions are merged in the conclusion's transition. This is possible only if s and s' are to some extent "complementary". Contrary to CCS, the rule (*Com*) can appear several times in the proof tree of a transition, because $s \bullet s'$ can still

contain virtual links (if s and s' had a virtual link in the same position) and can possibly be merged with other link chains. However, when $s \bullet s'$ is solid, no further synchronisation is possible (by Lemma 11 (ii)).

Example 24. Let $P = {}^{\tau}\backslash_{a}.P_1 \mid (\nu\, b)Q$ and $Q = {}^{b}\backslash_{\tau}.P_2 \mid {}^{a}\backslash_{b}.\mathbf{0}$, for some processes P_1 and P_2. The process ${}^{\tau}\backslash_{a}.P_1$ can perform an output on a, the process ${}^{b}\backslash_{\tau}.P_2$ can perform an input from b; the process ${}^{a}\backslash_{b}$ provides a one-shot link forwarder from a to b. Their links match along the sequence and a three-party interaction can take place. Together, these processes can indeed synchronise by agreeing to form the solid link chain ${}^{\tau}\backslash^{a}_{a}\backslash^{\tau}_{\tau}\backslash_{\tau}$ of length three, as follows.[3]

$$\dfrac{\dfrac{}{{}^{\tau}\backslash_{a}.P_1 \xrightarrow{{}^{\tau}\backslash^{\Box}_{a}\backslash^{\Box}_{\Box}\backslash_{\Box}} P_1}\,(Act) \qquad \dfrac{\dfrac{\dfrac{}{{}^{b}\backslash_{\tau}.P_2 \xrightarrow{{}^{\Box}\backslash^{\Box}_{\Box}\backslash^{b}_{\Box}\backslash_{\tau}} P_2}\,(Act) \qquad \dfrac{}{{}^{a}\backslash_{b}.\mathbf{0} \xrightarrow{{}^{\Box}\backslash^{a}_{\Box}\backslash^{\Box}_{b}\backslash_{\Box}} \mathbf{0}}\,(Act)}{Q \xrightarrow{{}^{\Box}\backslash^{a}_{\Box}\backslash^{b}_{b}\backslash_{\tau}} P_2|\mathbf{0}}\,(Com)}{(\nu\, b)Q \xrightarrow{{}^{\Box}\backslash^{a}_{\Box}\backslash^{\tau}_{\tau}\backslash_{\tau}} (\nu\, b)(P_2|\mathbf{0})}\,(Res)}{P \xrightarrow{{}^{\tau}\backslash^{a}_{a}\backslash^{\tau}_{\tau}\backslash_{\tau}} P_1|(\nu\, b)(P_2|\mathbf{0})}\,(Com)$$

The following lemma, whose proof goes by rule induction, shows that labels behave like an accordion. Concretely, any label s in a transition is replaceable with any other chain having a different number of virtual links ${}^{\Box}\backslash_{\Box}$ added to s according to the axioms of $\bowtie$. The result builds on the previous technical Lemmata 12 and 17. This fact will be later exploited in Section 5, where the abstract semantics is given.

Lemma 25 (Accordion Lemma)**.** *If $P \xrightarrow{s} P'$ and $s \bowtie s'$, then $P \xrightarrow{s'} P'$.*

Proof. The proof is by rule induction.

rule (Act) Assume $\ell.P \xrightarrow{s} P$ with $s \bowtie s'$. We need to prove that $\ell.P \xrightarrow{s'} P$ Since by hypothesis $s \bowtie \ell$ then, by transitivity, $s' \bowtie \ell$. By applying rule (Act) we get the thesis $\ell.P \xrightarrow{s'} P$.

[3]Note that b is restricted and therefore the matched communication on b is replaced by τ in the observed chain.

rule (Lsum) Assume $P + Q \xrightarrow{s} P'$ with $P \xrightarrow{s} P'$ and $s \bowtie s'$. We need to prove that $P + Q \xrightarrow{s'} P'$. By inductive hypothesis we have that $P \xrightarrow{s'} P'$, and by applying rule $(Lsum)$ we get the thesis. For the rules **(LPar)**, and **(Ide)** the proof is similar to this case and thus omitted.

rule (Res) Assume $(\nu\, a)P \xrightarrow{(\nu\, a)s} (\nu\, a)P'$ with $P \xrightarrow{s} P'$ and $(\nu\, a)s \bowtie s'$. We want to prove that $(\nu\, a)P \xrightarrow{s'} (\nu\, a)P'$. By Lemma 17, there exists s'' s.t. $s' = (\nu\, a)s''$ with $s \bowtie s''$. Then, by inductive hypothesis we have that $P \xrightarrow{s''} P'$, and by applying rule (Res) we obtain the thesis.

rule (Ren) Assume $P[\phi] \xrightarrow{s[\phi]} P'[\phi]$ with $P \xrightarrow{s} P'$ and $s[\phi] \bowtie s'$. We want to prove that $P[\phi] \xrightarrow{s'} P'[\phi]$. By Lemma 20, there exists s'' such that $s' = s''[\phi]$ and $s \bowtie s''$. Then, by inductive hypothesis we have that $P \xrightarrow{s''} P'$, and by applying rule (Ren) we get the thesis.

rule (Com) Assume $P|Q \xrightarrow{s} P'|Q'$ and $s \bowtie s'$. We want to prove that $P|Q \xrightarrow{s'} P'|Q'$. By hypothesis, there exist s_1 and s_2 such that $P \xrightarrow{s_1} P'$ and $Q \xrightarrow{s_2} Q'$, with $s = s_1 \bullet s_2$. By Lemma 12, there exist s_1' and s_2' such that $s_1' \bowtie s_1$, $s_2' \bowtie s_2$, and $s_1' \bullet s_2' = s'$. By inductive hypothesis, $P \xrightarrow{s_1'} P'$ and $Q \xrightarrow{s_2'} Q'$. Finally, by applying rule (Com) we get the thesis.

□

Example 26 (Forwarders). We give a few examples to show how flexible is CNA for defining "forwarding" policies. We have already seen a one-shot and one-hop forwarder from a to b that can be written as ${}^{a}\backslash_{b}.\mathbf{0}$. Its persistent version is just written as $R(a,b) \triangleq {}^{a}\backslash_{b}.R(a,b)$. Moreover, the process $R(a,b)\ |R(b,a)$ provides a sort of name fusion, making a and b interchangeable.

An alternating forwarder $A(a,b,c)$ from a to b first and then to c can be defined as

$$A(a,b,c) \triangleq {}^{a}\backslash_{b}.{}^{a}\backslash_{c}.A(a,b,c).$$

A persistent non-deterministic forwarder $C(a,\widetilde{c})$ (the C stands for *choice*), from a to $c_1, ..., c_n$ can be written as

$$C(a,\widetilde{c}) \triangleq {}^{a}\backslash_{c_1}.C(a,\widetilde{c}) + \cdots + {}^{a}\backslash_{c_n}.C(a,\widetilde{c}).$$

Similarly, $J(\widetilde{b}, a)$ defined as (the J stands for *join*[4])

$$J(\widetilde{b}, a) \triangleq {}^{b_1}\backslash_a.J(\widetilde{b}, a) + \cdots + {}^{b_m}\backslash_a.J(\widetilde{b}, a)$$

is a persistent non-deterministic forwarder, from $b_1, ..., b_m$ to a.

By combining the two processes above as in

$$F(\widetilde{b}, \widetilde{c}) \triangleq (\nu\, a)(J(\widetilde{b}, a) \mid C(a, \widetilde{c}))$$

we obtain a persistent forwarder from any of the b_i's to any of the c_j's. The only admissible transitions have indeed the form $F(\widetilde{b}, \widetilde{c}) \xrightarrow{s} F(\widetilde{b}, \widetilde{c})$ with $s \bowtie {}^{b_i}\backslash^{\tau}_{\tau}\backslash_{c_j}$ for some $i \in [1, m]$ and $j \in [1, n]$ (because interaction on a is restricted and $(\nu\, a)({}^{b_i}\backslash^{a}_{a}\backslash_{c_j}) = {}^{b_i}\backslash^{\tau}_{\tau}\backslash_{c_j}$).

Example 27 (Blind routing in CNA). We have already observed that CCS processes can be transformed to CNA processes just by transforming action prefixes in link prefixes. Correspondingly, the blind routing example from Section 2 (Example 1) can be encoded in CNA just as follows, where for readability we have omitted the parameters from definitions.

$$\begin{aligned}
A_i &\triangleq {}^{\tau}\backslash_{req_i}.{}^{\tau}\backslash_{think}.A_i \quad \text{for } i \in [1,2] \\
S_j &\triangleq {}^{srv_j}\backslash_{\tau}.{}^{\tau}\backslash_{exec}.S_j + {}^{\tau}\backslash_{busy}.{}^{\tau}\backslash_{\tau}.S_j \quad \text{for } j \in [1,2] \\
R &\triangleq {}^{req_1}\backslash_{\tau}.({}^{\tau}\backslash_{srv_1}.R + {}^{\tau}\backslash_{srv_2}.R) + {}^{req_2}\backslash_{\tau}.{}^{\tau}\backslash_{srv_2}.R \\
N &\triangleq (\nu\, \widetilde{req})(\nu\, \widetilde{srv})(A_1 \mid A_2 \mid R \mid S_1 \mid S_2)
\end{aligned}$$

For example, the following transitions can be derived from the SOS rules accounting for the case where a request from A_1 is assigned to S_2:

$$\begin{aligned}
N &\xrightarrow{{}^{\tau}\backslash^{\tau}_{\tau}\backslash_{\tau}} (\nu\, \widetilde{req})(\nu\, \widetilde{srv})({}^{\tau}\backslash_{think}.A_1 \mid A_2 \mid ({}^{\tau}\backslash_{srv_1}.R + {}^{\tau}\backslash_{srv_2}.R) \mid S_1 \mid S_2) \\
&\xrightarrow{{}^{\tau}\backslash^{\tau}_{\tau}\backslash_{\tau}} (\nu\, \widetilde{req})(\nu\, \widetilde{srv})({}^{\tau}\backslash_{think}.A_1 \mid A_2 \mid R \mid S_1 \mid {}^{\tau}\backslash_{exec}.S_2) \\
&\xrightarrow{{}^{\tau}\backslash_{think}} (\nu\, \widetilde{req})(\nu\, \widetilde{srv})(A_1 \mid A_2 \mid R \mid S_1 \mid {}^{\tau}\backslash_{exec}.S_2) \\
&\xrightarrow{{}^{\tau}\backslash_{exec}} N
\end{aligned}$$

[4]With the term "join" we refer to the fact that messages from different sources are forwarded to the same channel. It has no relation with join patterns in join calculus.

Note that, in the first two steps, interactions on channels req_1 and srv_2 are restricted and thus not observable, in fact $(\nu\, req_1)({}^{\tau}\backslash^{req_1}_{req_1}\backslash_{\tau}) = {}^{\tau}\backslash^{\tau}_{\tau}\backslash_{\tau}$ and $(\nu\, srv_2)({}^{\tau}\backslash^{srv_2}_{srv_2}\backslash_{\tau}) = {}^{\tau}\backslash^{\tau}_{\tau}\backslash_{\tau}$.

Example 28 (Acknowledged routing in CNA). The features of CNA become evident in our running example when we come to modelling acknowledged routing (Example 2). This is because the explicit acknowledgment is no longer necessary as it can be implicitly accounted for by the ability to construct a chain of links.[5] Correspondingly, we set

$$\begin{aligned}
A_i &\triangleq {}^{\tau}\backslash_{req_i}.{}^{\tau}\backslash_{think}.A_i \quad \text{for } i \in [1,2] \\
S_j &\triangleq {}^{srv_j}\backslash_{\tau}.{}^{\tau}\backslash_{exec}.S_j + {}^{\tau}\backslash_{busy}.{}^{\tau}\backslash_{\tau}.S_j \quad \text{for } j \in [1,2] \\
R &\triangleq {}^{req_1}\backslash_{srv_1}.R + {}^{req_1}\backslash_{srv_2}.R + {}^{req_2}\backslash_{srv_2}.R \\
M &\triangleq (\nu\, \widetilde{req})(\nu\, \widetilde{srv})(A_1 \mid A_2 \mid R \mid S_1 \mid S_2)
\end{aligned}$$

Note that channels ack_i are not needed and CNA processes A_i and S_j are defined as in the case of blind routing; only the syntax of R has been changed to link, in one single step, the requests from agents with the availability of servers.

For example, the following transitions can be derived from the SOS rules accounting for the case where a request from A_1 is assigned to S_2:

$$\begin{aligned}
M &\xrightarrow{{}^{\tau}\backslash^{\tau}_{\tau}\backslash^{\tau}_{\tau}\backslash_{\tau}} (\nu\, \widetilde{req})(\nu\, \widetilde{srv})({}^{\tau}\backslash_{think}.A_1 \mid A_2 \mid R \mid S_1 \mid {}^{\tau}\backslash_{exec}.S_2) \\
&\xrightarrow{{}^{\tau}\backslash_{think}} (\nu\, \widetilde{req})(\nu\, \widetilde{srv})(A_1 \mid A_2 \mid R \mid S_1 \mid {}^{\tau}\backslash_{exec}.S_2) \\
&\xrightarrow{{}^{\tau}\backslash_{exec}} M
\end{aligned}$$

Note that, in the first step, the interaction on channels req_1 and srv_2 is restricted and thus not observable, in fact $(\nu\, req_1)(\nu\, srv_2)({}^{\tau}\backslash^{req_1}_{req_1}\backslash^{srv_2}_{srv_2}\backslash_{\tau}) = {}^{\tau}\backslash^{\tau}_{\tau}\backslash^{\tau}_{\tau}\backslash_{\tau}$.

Example 29 (Composite, acknowledged routing in CNA). Notably, the infrastructure presented in the previous example is already designed in a mod-

[5] Of course, one could also just encode the CCS processes for acknowledged routing just by translating their prefixes as we have done for blind routing.

ular way: infrastructures can be composed without requiring any change and no dead path detection or backtracking mechanisms have to be put in place, because they are taken care by the CNA "middleware".

For instance, take the CNA versions of the infrastructures presented in Example 3:

$$\begin{aligned} R' &\triangleq {}^{req_1}\backslash_{s_1}.R' + {}^{req_1}\backslash_{s_2}.R' + {}^{req_2}\backslash_{s_2}.R' \\ R'' &\triangleq {}^{s_1}\backslash_{s'_1}.R'' + {}^{s_2}\backslash_{s'_2}.R' \\ R''' &\triangleq {}^{s'_2}\backslash_{srv_2}.R''' \\ R &\triangleq (\nu\, \widetilde{s'})(\nu\, \widetilde{s})(R' \mid R'' \mid R''') \end{aligned}$$

Besides looking considerably more concise than their CCS versions, they make evident that the delivery of any request to a server is atomic as any process executes one (link) action and reduces (recursively) to itself.

At a closer inspection, one may notice that the only possible transitions for R are the following ones (up to $\blacktriangleright\!\!\blacktriangleleft$, as explained by the Accordion Lemma 25):

$$R \xrightarrow{{}^{req_1}\backslash^{\tau}_{\tau}\backslash^{\tau}_{\tau}\backslash_{srv_2}} R \quad \text{and} \quad R \xrightarrow{{}^{req_2}\backslash^{\tau}_{\tau}\backslash^{\tau}_{\tau}\backslash_{srv_2}} R.$$

As in the previous examples, note that, e.g. in the first step, the interaction on channels s_2 and s'_2 is restricted and thus not observable, in fact $(\nu\, s_2)(\nu\, s'_2)^{req_1}\backslash^{s_2}_{s_2}\backslash^{s'_2}_{s'_2}\backslash_{srv_2} = {}^{req_1}\backslash^{\tau}_{\tau}\backslash^{\tau}_{\tau}\backslash_{srv_2}$.

Therefore, at a suitable level of abstraction, in which routing details are omitted, we would like to relate the composite infrastructure R with the monolithic infrastructure R_m defined as follows:

$$R_m \triangleq {}^{req_1}\backslash_{srv_2}.R_m + {}^{req_2}\backslash_{srv_2}.R_m$$

In the next section we show how this can be formalised.

5. Abstract semantics

As usual, we can use the LTS semantics to define suitable behavioural equivalences over processes. We focus on bisimulation relations. The accordion lemma

(Lemma 25) implies that it makes no sense to distinguish between two labels s and s' such that $s \bowtie s'$, because if one process p can do s and reach p', then it can also do s' and still reach p'. However, when comparing two labels we would like to abstract away also from the number of hops performed and from the size (not just the length) of the chains, as the following example suggests.

Example 30. Take the one-hop forwarder $R(a,b) \triangleq {}^{a}\backslash_{b}.R(a,b)$. From the operational semantics it is immediate to check that its transitions are all and only $R(a,b) \xrightarrow{s} R(a,b)$ such that $s \bowtie {}^{a}\backslash_{b}$.

Now connect together two one-hop forwarders in a sequence to form the routing infrastructure $T(a,b) \triangleq (\nu\, c)(R(a,c) \mid R(c,b))$. Again it is immediate to check that its transitions are all and only $T(a,b) \xrightarrow{s} T(a,b)$ such that $s \bowtie {}^{a}\backslash^{\tau}_{\tau}\backslash_{b}$.

Intuitively, we would like $R(a,b)$ and $T(a,b)$ to be interchangeable, as they offer the same routing service. If we were to compare $R(a,b)$ and $T(a,b)$ using plain bisimilarity, where labels must be matched syntactically, then they would not be equivalent. Also if we relax bisimulation by matching transition labels up-to $\bowtie$, the two terms are not equivalent, as it is not true that ${}^{a}\backslash_{b} \bowtie {}^{a}\backslash^{\tau}_{\tau}\backslash_{b}$ (they have different sizes, i.e. they have a different number of solid links, and size is preserved by $\bowtie$).

To define a bisimilarity equivalence that relates processes such as $R(a,b)$ and $T(a,b)$ above, we introduce an equivalence coarser than $\bowtie$, written $\rhd\lhd$, that we use to match labels in the bisimulation game, according to which e.g. ${}^{a}\backslash_{b} \rhd\lhd {}^{a}\backslash^{\tau}_{\tau}\backslash_{b}$.

The only difference between $\rhd\lhd$ and $\bowtie$ is that the additional axiom of $\rhd\lhd$ abstracts away from intermediate matched actions that are silent. The intuition is that such matched actions cannot be split and used to compose longer chains, because they are silent and therefore the matching was made on a restricted channel.

Remark 31. We invite the reader to check that the Accordion Lemma 25 is only concerned with $\bowtie$ and not with $\rhd\lhd$.

We now consider the equivalences classes given by $\rtimes\ltimes$ and its representatives.

Definition 32 (Essential chain). A link chain is *essential* if it is composed by alternating solid and virtual links, with solid links at its extremes.

For example, the chain ${}^{a}\backslash^{\tau}_{\tau}\backslash^{b}_{b}\backslash_{c}$ is not essential, while the chain ${}^{a}\backslash^{\Box}_{b}\backslash^{b}_{\Box}\backslash_{c}$ is essential and we have ${}^{a}\backslash^{\tau}_{\tau}\backslash^{b}_{b}\backslash_{c} \rtimes\ltimes {}^{a}\backslash^{\Box}_{b}\backslash^{b}_{\Box}\backslash_{c}$.

The following lemma shows that each $\rtimes\ltimes$-equivalence class has a unique essential representative.

Lemma 33. *All of the following properties hold for any link chain s.*

(i) There exists an essential link chain s' such that $s \rtimes\ltimes s'$.

(ii) If s is essential, for any essential s' such that $s \rtimes\ltimes s'$, then $s = s'$.

It is immediate to check that by orienting the axioms in Definitions 8 and **??** from left to right we have a procedure for transforming any link chain s to a unique essential link chain s' such that $s \rtimes\ltimes s'$. We write $\mathsf{e}(s)$ to denote such a unique representative.

Corollary 34. *For any link chains s, s' we have $s \rtimes\ltimes s'$ iff $\mathsf{e}(s) = \mathsf{e}(s')$.*

The following properties are useful in the proof of the main result, i.e. the congruence property of our notion of bisimilarity (Theorem 42).

The first lemma says that the restriction operator respects the relation $\rtimes\ltimes$, in the sense that if one link chain s can be restricted in a then any chain $s' \rtimes\ltimes s$ can be extended to some $s'' \blacktriangleright\!\!\blacktriangleleft s'$ where a is matched and can be restricted.

Lemma 35. *If $s \rtimes\ltimes s'$, then for any a such that $(\nu\, a)s \neq \bot$ there exists $s'' \blacktriangleright\!\!\blacktriangleleft s'$ such that $(\nu\, a)s'' \neq \bot$ and $(\nu\, a)s \rtimes\ltimes (\nu\, a)s''$.*

The second lemma says that taken two link chains s_1 and s_1' in the same equivalence class, and given any link chain s_2 that can be merged with s_1, then it is possible to find a link chain s_2' (which is a stretched version of s_2) such that it can be merged with another link chain s_1'' (which is a stretched version of s_1') with the result being equivalent to $s_2 \bullet s_1$. This is graphically rendered in Figure 6.

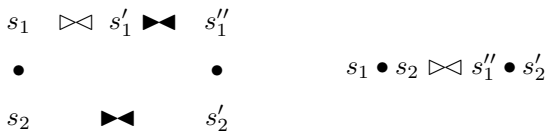


Figure 6: Graphic representation of Lemma 36.

Lemma 36. *If $s_1 \rhd\lhd s_1'$, then for any s_2 such that $s_2 \bullet s_1 \neq \bot$ there exist two link chains $s_2' \blacktriangleright\blacktriangleleft s_2$ and $s_1'' \blacktriangleright\blacktriangleleft s_1'$ such that $s_2' \bullet s_1'' \neq \bot$ and $s_2 \bullet s_1 \rhd\lhd s_2' \bullet s_1''$.*

Also the next lemma is introduced to prove the main Theorem 42.

Lemma 37. *Let s and s' be two link chains such that $s \rhd\lhd s'$, then for any renaming function $s[\phi] \rhd\lhd s'[\phi]$.*

Definition 38 (Network bisimulation). A *network bisimulation* $\mathbf{R}$ is a binary relation over CNA processes such that, if $P \,\mathbf{R}\, Q$ then:

- if $P \xrightarrow{s} P'$, then $\exists\ s'$, Q' such that $s' \rhd\lhd s$, $Q \xrightarrow{s'} Q'$, and $P' \,\mathbf{R}\, Q'$;
- if $Q \xrightarrow{s} Q'$, then $\exists\ s'$, P' such that $s' \rhd\lhd s$, $P \xrightarrow{s'} P'$, and $P' \,\mathbf{R}\, Q'$.

Note that, by Corollary 34, the requirement $s' \rhd\lhd s$ amounts just to checking that $\mathsf{e}(s') = \mathsf{e}(s)$.

Definition 39 (Network bisimilarity $\overset{\rhd\lhd}{\sim}$). We let $\overset{\rhd\lhd}{\sim}$ denote the largest network bisimulation and we say that P is *network bisimilar* to Q if $P \overset{\rhd\lhd}{\sim} Q$.

It can be shown that network bisimulations are closed with respect to union and composition and that $\overset{\rhd\lhd}{\sim}$ is an equivalence relation.

Example 40. Take the recursively defined processes $R(a,b) \triangleq {}^{a}\backslash_{b}.R(a,b)$ and $T(a,b) \triangleq (\nu\, c)(R(a,c) \mid R(c,a))$ from Example 30. It is straightforward to check that the relation

$$\mathbf{R} \triangleq \{(R(a,b), T(a,b))\}$$

is a network bisimulation, because ${}^{a}\backslash_{b} \rhd\lhd {}^{a}\backslash_{\tau}^{\tau}\backslash_{b}$, hence $R(a,b)$ and $T(a,b)$ are network bisimilar.

Example 41. Consider the two processes $P \triangleq {}^{a}\backslash_{b}.P$ and $Q \triangleq (\nu\, c)({}^{a}\backslash_{c} \mid {}^{c}\backslash_{b}.Q)$. We have that whenever $P \xrightarrow{s} P'$, then $P' = P$ and $\mathsf{e}(s) = {}^{a}\backslash_{b}$. Similarly, whenever $Q \xrightarrow{s} Q'$, then $Q' = (\nu\, c)(\mathbf{0}|Q)$ and $\mathsf{e}(s) = {}^{a}\backslash_{b}$. Then we prove that $P \mathrel{\bowtie\!\!\!\sim} Q$ by showing that the relation $\mathbf{R}$ below:

$$\mathbf{R} \triangleq \{(P, R) \mid \exists n.R = C^n[Q]\}$$

is a network bisimulation, where $C^n[Q]$ is inductively defined by letting $C^0[Q] \triangleq Q$ and $C^{n+1}[Q] \triangleq C[C^n[Q]]$ for $C[\cdot]$ the context $(\nu\, c)(\mathbf{0}|\cdot)$. Intuitively, the context C^n mimics the effects of n internal interactions in Q, as any internal interaction in Q leaves a zero process in parallel with Q. The thesis simply follows by noting that, for any n, whenever $C^n[Q] \xrightarrow{s} R'$ then $R' = C^{n+1}[Q]$ and $\mathsf{e}(s) = {}^{a}\backslash_{b}$: by induction on n, the base case $C^0[Q] = Q$ has been already observed above, while the inductive case, where we consider $C^{n+1}[Q] = (\nu\, c)(\mathbf{0}|C^n[Q])$, follows immediately from the inductive hypothesis.

We are now ready to prove the first main result, i.e. that network bisimilarity is preserved by all the operators of CNA.

Theorem 42. *Network bisimilarity is a congruence.*

Proof. We show that network bisimilarity is preserved by all the operators. The proof uses standard arguments. The interesting cases are that of restriction, renaming and parallel composition, the others are just suitable rephrasing of the corresponding proofs in the CCS literature.

Prefix We want to prove that if $P \mathrel{\bowtie\!\!\!\sim} Q$ then for any ℓ we have $\ell.P \mathrel{\bowtie\!\!\!\sim} \ell.Q$. We define the relation $\mathbf{R}_{pre} \triangleq \{(\ell.P, \ell.Q) \mid P \mathrel{\bowtie\!\!\!\sim} Q\} \cup \mathrel{\bowtie\!\!\!\sim}$ and show that $\mathbf{R}_{pre}$ is a network bisimulation. The case when $(P, Q) \in \mathrel{\bowtie\!\!\!\sim}$ is obvious. Take $(\ell.P, \ell.Q) \in \{(\ell.P, \ell.Q) \mid P \mathrel{\bowtie\!\!\!\sim} Q\}$. If $\ell.P \xrightarrow{s} P$ with $s \blacktriangleright\!\!\blacktriangleleft \ell$ then also $\ell.Q \xrightarrow{s} Q$ and $P\ \mathbf{R}_{pre}\ Q$ as $P \mathrel{\bowtie\!\!\!\sim} Q$. Vice versa, if $\ell.Q \xrightarrow{s} Q$ with $s \blacktriangleright\!\!\blacktriangleleft \ell$ then also $\ell.P \xrightarrow{s} P$ and $P\ \mathbf{R}_{pre}\ Q$ as $P \mathrel{\bowtie\!\!\!\sim} Q$.

Restriction We want to prove that if $P \mathrel{\bowtie\!\!\!\sim} Q$ then for any a we have $(\nu\, a)P \mathrel{\bowtie\!\!\!\sim} (\nu\, a)Q$. We let $\mathbf{R}_{res} \triangleq \{((\nu\, a)P, (\nu\, a)Q) | P \mathrel{\bowtie\!\!\!\sim} Q\}$ and show that $\mathbf{R}_{res}$ is

a network bisimulation. Suppose $P \overset{\bowtie}{\sim} Q$ and $(\nu\, a)P \xrightarrow{(\nu\, a)s} (\nu\, a)P'$, for some s and P' such that $P \xrightarrow{s} P'$. By assumption, we know that $P \overset{\bowtie}{\sim} Q$ and therefore there exist s', Q' such that $Q \xrightarrow{s'} Q'$ with $\mathsf{e}(s') = \mathsf{e}(s)$ and $P' \overset{\bowtie}{\sim} Q'$. By Lemma 35, there exists $s'' \blacktriangleright\!\blacktriangleleft s'$ such that $(\nu\, a)s'' \neq \bot$ and $(\nu\, a)s'' \bowtie (\nu\, a)s$. Hence $(\nu\, a)Q \xrightarrow{(\nu\, a)s''} (\nu\, a)Q'$ and, by definition of $\mathbf{R}_{res}$, we obtain $(\nu\, a)P' \; \mathbf{R}_{res} \; (\nu\, a)Q'$.

Renaming We want to prove that if $P \overset{\bowtie}{\sim} Q$ then for any renaming function ϕ we have $P[\phi] \overset{\bowtie}{\sim} Q[\phi]$. Let $\mathbf{R}_{ren} \triangleq \{(P[\phi], Q[\phi]) \mid P \overset{\bowtie}{\sim} Q\}$ and show that $\mathbf{R}_{ren}$ is a network bisimulation. Suppose $P \overset{\bowtie}{\sim} Q$ and $P[\phi] \xrightarrow{s} P'[\phi]$, for some s, P'. By rule (Ren), it must exist s' such that $s = s'[\phi]$, and $P \xrightarrow{s'} P'$. By assumption we know that $P \overset{\bowtie}{\sim} Q$ and therefore there exist s'', and Q' s.t. $Q \xrightarrow{s''} Q'$ with $\mathsf{e}(s') = \mathsf{e}(s'')$. By Lemma 37, we have $\mathsf{e}(s'[\phi]) = \mathsf{e}(s''[\phi])$. Hence $Q[\phi] \xrightarrow{s''[\phi]} Q'[\phi]$ and, by definition of $\mathbf{R}_{ren}$, we obtain $P[\phi] \; \mathbf{R}_{ren} \; Q[\phi]$.

Choice We want to prove that if $P \overset{\bowtie}{\sim} Q$ then for any R we have $P+R \overset{\bowtie}{\sim} Q+R$. We define the relation $\mathbf{R}_{sum} \triangleq \{(P+R, Q+R) \mid P \overset{\bowtie}{\sim} Q\} \cup \overset{\bowtie}{\sim}$ and show that $\mathbf{R}_{sum}$ is a network bisimulation. The case when $(P, Q) \in \overset{\bowtie}{\sim}$ is immediate. Take $(P+R, Q+R) \in \{(P+R, Q+R) \mid P \overset{\bowtie}{\sim} Q\}$. Suppose $P + R \xrightarrow{s} T$. We want to prove that $Q + R \xrightarrow{s'} T'$ with $\mathsf{e}(s) = \mathsf{e}(s')$, and $T \; \mathbf{R}_{sum} \; T'$. There are two cases to be considered, depending on the last SOS rule used, to prove $P + R \xrightarrow{s} T$. If the last used rule is

- *(Rsum)*, then it means that $R \xrightarrow{s} R'$ for some R' with $T = R'$. But then, by using $(Rsum)$, we have $Q + R \xrightarrow{s} T'$, with $T' = R'$ and $T = R' \; \mathbf{R}_{sum} \; R' = T'$ by reflexivity of network bisimilarity $\overset{\bowtie}{\sim} \subseteq \mathbf{R}_{sum}$.
- *(Lsum)*, then it means that $P \xrightarrow{s} P'$ for some P' with $T = P'$. By assumption, we know that $P \overset{\bowtie}{\sim} Q$ and therefore there exist s' and Q' such that $Q \xrightarrow{s'} Q'$ with $\mathsf{e}(s) = \mathsf{e}(s')$ and $P' \overset{\bowtie}{\sim} Q'$. By applying the rule $(Lsum)$, we obtain that $Q + R \xrightarrow{s'} Q'$ and we are done.

Parallel We want to prove that if $P \stackrel{\bowtie}{\sim} Q$ then for any R we have $P|R \stackrel{\bowtie}{\sim} Q|R$. We define the relation $\mathbf{R}_{par} \triangleq \{(P|R, Q|R) \mid P \stackrel{\bowtie}{\sim} Q\}$ and show that $\mathbf{R}_{par}$ is a network bisimulation. Suppose $P \stackrel{\bowtie}{\sim} Q$ and $P|R \stackrel{s}{\rightarrow} T$. We want to prove that $Q|R \stackrel{s}{\rightarrow} T'$ with $T \ \mathbf{R}_{par} \ T'$. There are three cases to be considered, depending on the last SOS rule used to prove $P|R \stackrel{s}{\rightarrow} T$. If the last used rule is

- *(Rpar)*, then it means that $R \stackrel{s}{\rightarrow} R'$ for some R' with $T = P|R'$. But then, by using *(Rpar)*, we have $Q|R \stackrel{s}{\rightarrow} Q|R'$ and $P|R' \ \mathbf{R}_{par} \ Q|R'$, by definition of $\mathbf{R}_{par}$.
- *(Lpar)*, then it means that $P \stackrel{s}{\rightarrow} P'$ for some P' with $T = P'|R$. By assumption, we know that $P \stackrel{\bowtie}{\sim} Q$ and therefore there exist s', Q' such that $Q \stackrel{s'}{\rightarrow} Q'$ with $\mathsf{e}(s) = \mathsf{e}(s')$ and $P' \stackrel{\bowtie}{\sim} Q'$. By applying the rule *(Lpar)*, we have that $Q|R \stackrel{s'}{\rightarrow} Q'|R$ and we are done because $P'|R \ \mathbf{R}_{par} \ Q'|R$, by definition of $\mathbf{R}_{par}$.
- *(Com)*, then it means that $P \stackrel{s_1}{\rightarrow} P'$, $R \stackrel{s_2}{\rightarrow} R'$, for some s_1, s_2, P', R' with $s = s_1 \bullet s_2$ and $T = P'|R'$. By assumption, we know that $P \stackrel{\bowtie}{\sim} Q$ and therefore there exist s_1', Q' such that $Q \stackrel{s_1'}{\rightarrow} Q'$ with $\mathsf{e}(s_1) = \mathsf{e}(s_1')$ and $P' \stackrel{\bowtie}{\sim} Q'$. Now it may be the case that $s_1' \bullet s_2$ is not defined, but by Lemma 36, we know that s_1' and s_2 can be stretched respectively to $s_1'' \blacktriangleright\!\blacktriangleleft s_1'$ and $s_2' \blacktriangleright\!\blacktriangleleft s_2$ by inserting or removing enough virtual links to have that $s_1'' \bullet s_2'$ is defined and $\mathsf{e}(s_1 \bullet s_2) = \mathsf{e}(s_1'' \bullet s_2')$. By the Accordion Lemma 25, $Q \stackrel{s_1''}{\rightarrow} Q'$ and $R \stackrel{s_2'}{\rightarrow} R'$. We conclude by applying rule *(Com)*: $Q|R \stackrel{s'}{\rightarrow} Q'|R'$ with $s' = s_1'' \bullet s_2' \bowtie s$ and $P'|R' \ \mathbf{R}_{par} \ Q'|R'$, by definition of $\mathbf{R}_{par}$.

Recursion Let E and F be two processes that invoke the process identifier X. Let us denote with $E\{P/X\}$ the process obtained by replacing in E every occurrence of the identifier X with the process P. Assume that for any process P we have $E\{P/X\} \stackrel{\bowtie}{\sim} F\{P/X\}$. We want to prove that, given the process definitions $A \triangleq E\{A/X\}$, $B \triangleq F\{B/X\}$, then $A \stackrel{\bowtie}{\sim} B$. The proof proceeds by showing that:

1. If $A \triangleq Q$ is a process definition, then $A \mathrel{\bowtie\!\!\!\sim} Q$.
2. Given the process definitions $A \triangleq E\{A/X\}$ and $B \triangleq F\{B/X\}$, for any process G that invokes X we have $G\{A/X\} \mathrel{\bowtie\!\!\!\sim} G\{B/X\}$.

Then, we have $A \mathrel{\bowtie\!\!\!\sim} E\{A/X\} \mathrel{\bowtie\!\!\!\sim} E\{B/X\} \mathrel{\bowtie\!\!\!\sim} F\{B/X\} \mathrel{\bowtie\!\!\!\sim} B$. The proof of (1) is immediate by rule *(Ide)*, as A and Q have exactly the same transitions, while the proof of (2) proceeds in the standard way exploiting induction on derivations, as detailed in the appendix.

□

Remark 43. As for CCS, it can be proved that several useful axioms over processes hold up to network bisimilarity, like the commutative monoidal laws for $|$ and $+$, the idempotence of $+$ and the usual laws about restriction.

5.1. Semantics Closure with Respect to Substitutions

One relevant difference w.r.t. strong and weak bisimilarity in CCS is that network bisimilarity is also closed with respect to substitutions.

At the level of chains, name substitution is defined as the renaming (see Definition 18). Not to overload the notation, we denote substitution with $\{-/-\}$.

Given $s = \ell_1 \dots \ell_n$, with $\ell_i = {}^{\alpha_1}\backslash_{\beta_i}$ for $i \in [1, n]$, we define the substitution of channel a with channel b in a link chain s, $s\{b/a\}$ as follows

$$s\{b/a\} = \ell_1\{b/a\} \dots \ell_n\{b/a\}$$

$$\ell_i\{b/a\} = {}^{\alpha_i\{b/a\}}\backslash_{\beta_i\{b/a\}} \qquad \alpha\{b/a\} = \begin{cases} b & \text{if } \alpha = a \\ \alpha & \text{otherwise} \end{cases}$$

The first observation is that equivalence $\bowtie$ (but also $\blacktriangleright\!\!\blacktriangleleft$) is closed with respect to substitution, as stated by the next lemma.

Lemma 44. *For any a, b, s, s'*

(i) If $s \blacktriangleright\!\!\blacktriangleleft s'$ then $s\{b/a\} \blacktriangleright\!\!\blacktriangleleft s'\{b/a\}$.

(ii) If $s \bowtie s'$ then $s\{b/a\} \bowtie s'\{b/a\}$.

Next, we prove that transitions are respected by substitutions. On processes, name substitution differs from renaming and a different notation is used. Let us denote by $P\{b/a\}$ the simultaneous, capture-avoiding substitution of all the (free) occurrences of a with b in P. Substitution is defined inductively as follows.

$$\begin{aligned}
\mathbf{0}\{b/a\} &\triangleq \mathbf{0} \\
\ell.P\{b/a\} &\triangleq \ell\{b/a\}.P\{b/a\} \\
(P+Q)\{b/a\} &\triangleq (P\{b/a\}) + (Q\{b/a\}) \\
(P|Q)\{b/a\} &\triangleq (P\{b/a\})|(Q\{b/a\}) \\
((\nu c)P)\{b/a\} &\triangleq (\nu d)(P\{d/c\}\{b/a\}) \text{ with } d \text{ fresh} \\
P[\phi]\{b/a\} &\triangleq P\{\phi^{-1}(b)/\phi^{-1}(a)\}[\phi] \\
A(\tilde{c})\{b/a\} &\triangleq A(\tilde{c}\{b/a\})
\end{aligned}$$

Substitution enjoys properties similar to that of renaming. In particular, in the proof of Lemma 46 we exploit the following property (see Lemma 19(ii)), whose proof follows immediately by definition of $\bullet$ and substitution.

Lemma 45. *For any a, b, s_1, s_2, if $s_1 \bullet s_2$ is defined, then $(s_1 \bullet s_2)\{b/a\} = s_1\{b/a\} \bullet (s_2\{b/a\})$.*

Lemma 46. *For any process P the following holds:*

1. *if $P \xrightarrow{s} P'$ then $P\{b/a\} \xrightarrow{s\{b/a\}} P'\{b/a\}$.*
2. *if $P\{b/a\} \xrightarrow{s} P'$ then there exists s' and P'' such that $P \xrightarrow{s'} P''$ with $P' = P''\{b/a\}$ and $s \bowtie s'\{b/a\}$.*

Proof. We prove the two items separately.

1. The proof is straightforward by rule induction and thus omitted.
2. The proof is in two steps. First we observe that whenever $P\{b/a\} \xrightarrow{s} P'$ then, by the Accordion Lemma 25, $P\{b/a\} \xrightarrow{s''} P'$ with $s \bowtie s''$ where there is no matched occurrences of b in s'' (the chain s'' is obtained from s by applying the fourth axiom in Definition 8 as many times as needed). Then we prove that if $P\{b/a\} \xrightarrow{s''} P'$ where there is no matched

occurrences of b in s'', then there exists s' and P'' such that $P \xrightarrow{s'} P''$ with $P' = P''\{b/a\}$ and $s'' = s'\{b/a\}$. The rule proceeds by rule induction as detailed below.

Rule (Act) By hypothesis, $P = \ell.P''$, and we get $P\{b/a\} = (\ell.P'')\{b/a\} = \ell\{b/a\}.P''\{b/a\}$ with $P\{b/a\} \xrightarrow{s''} P''\{b/a\}$ and $s'' \bowtie \ell\{b/a\}$. Moreover, by rule (Act), $P = \ell.P'' \xrightarrow{s'} P''$ for any $s' \bowtie \ell$. In particular, we can choose s' to have the same virtual links (and in the same positions) as s'', so that $s'\{b/a\} = s''$. By putting $P' = P''\{b/a\}$, we are done.

Rule (Res) By hypothesis, $P = (\nu\, c)Q$ and without loss of generality assume that $c \neq a, b$. We get $P\{b/a\} = ((\nu\, c)Q)\{b/a\} = (\nu\, c)(Q\{b/a\})$. Hence, there exist Q' and s_1'' such that

$$P\{b/a\} = (\nu\, c)(Q\{b/a\}) \xrightarrow{(\nu\, c)s_1''} (\nu\, c)Q' = P'$$

with $s'' = (\nu\, c)s_1''$ and $Q\{b/a\} \xrightarrow{s_1''} Q'$. Since there is no matched occurrence of b in s'', there is none in s_1''. By inductive hypothesis, there exist Q'' and s_1' such that $Q \xrightarrow{s_1'} Q''$, with $s_1'\{b/a\} = s_1''$, and $Q' = Q''\{b/a\}$. Since $(\nu\, c)s_1'' = (\nu\, c)(s_1'\{b/a\})$ is defined then also $(\nu\, c)s_1'$ is defined, because $c \neq a, b$. Thus, we can apply rule (Res) to get $(\nu\, c)Q \xrightarrow{(\nu\, c)s_1'} (\nu\, c)Q''$, and we take $P'' = (\nu\, c)Q''$ and $s' = (\nu\, c)s_1'$. Then we get $P' = (\nu\, c)Q' = (\nu\, c)(Q''\{b/a\}) = P''\{b/a\}$ and $s'' = (\nu\, c)s_1'' = (\nu\, c)(s_1'\{b/a\}) = ((\nu\, c)s_1')\{b/a\} = s'\{b/a\}$.

Rule (Com) By hypothesis, $P = R|Q$ and, by rule (Com), we have

$$P\{b/a\} = (R|Q)\{b/a\} = (R\{b/a\}|Q\{b/a\}) \xrightarrow{s_1'' \bullet s_2''} (R'|Q') = P'$$

with $s'' = s_1'' \bullet s_2''$, $R\{b/a\} \xrightarrow{s_1''} R'$ and $Q\{b/a\} \xrightarrow{s_2''} Q'$. Since all there is no matched occurrence of b in s'', there is none in both s_1'' and s_2''. By inductive hypothesis, there exist R'', s_1', Q'', s_2' such that $R \xrightarrow{s_1'} R''$ and $Q \xrightarrow{s_2'} Q''$, with $s_1'' = s_1'\{b/a\}$, $R' = R''\{b/a\}$, $s_2'' = s_2'\{b/a\}$, $Q' = Q''\{b/a\}$. Now we observe that $s_1' \bullet s_2'$ is defined. In fact the

only reason for which $s'_1 \bullet s'_2$ is undefined when $s'_1\{b/a\} \bullet (s'_2\{b/a\})$ is defined would be that an action a should be paired with an action b before the substitution takes place, but this is ruled out by the assumption that there is no matched occurrence of b in s''. By rule (Com), we have $R|Q \xrightarrow{s'_1 \bullet s'_2} R''|Q''$. Now, we take $P'' = R''|Q''$, $s' = s'_1 \bullet s'_2$ and we get $P' = R'|Q' = R''\{b/a\}|Q''\{b/a\} = (R''|Q'')\{b/a\} = P''\{b/a\}$, $s'' = s''_1 \bullet s''_2 = s'_1\{b/a\} \bullet (s'_2\{b/a\}) = (s'_1 \bullet s'_2)\{b/a\} = s'\{b/a\}$ and we are done.

Rule (Ren) By hypothesis, $P = Q[\phi]$. Let $a' = \phi^{-1}(a)$ and $b' = \phi^{-1}(b)$. We get that $P\{b/a\} = Q[\phi]\{b/a\} = Q\{b'/a'\}[\phi]$ and, by rule (Ren) there exist Q' and s''_1 such that

$$P\{b/a\} = Q[\phi]\{b/a\} = Q\{b'/a'\}[\phi] \xrightarrow{s''_1[\phi]} Q'[\phi] = P'$$

with $s'' = s''_1[\phi]$ and $Q\{b'/a'\} \xrightarrow{s_1} Q'$. Since there is no matched occurrence of b in s'' it must be the case that there is no matched occurrence of b' in s''_1. By inductive hypothesis, there exists Q'' and s'_1 s.t. $Q \xrightarrow{s'_1} Q''$, with $s''_1 = s'_1\{b'/a'\}$, and $Q' = Q''\{b'/a'\}$. By rule (Ren), $Q[\phi] \xrightarrow{s'_1[\phi]} Q''[\phi]$, and we take $P'' = Q''[\phi]$ and $s' = s'_1[\phi]$. Then we get $P' = Q'[\phi] = Q''\{b'/a'\}[\phi] = Q''[\phi]\{b/a\} = P''\{b/a\}$ and $s'' = s''_1[\phi] = s'_1\{b'/a'\}[\phi] = s'_1[\phi]\{b/a\} = s'\{b/a\}$.

For the remaining rules the proofs are simpler and thus omitted.

□

Proposition 47. *For any processes P, Q, if $P \overset{\bowtie}{\sim} Q$ then for any channel a, b we have $P\{b/a\} \overset{\bowtie}{\sim} Q\{b/a\}$.*

Proof. We define the relation $\mathbf{R}_{sub} = \{(P\{b/a\}, Q\{b/a\}) \mid P \overset{\bowtie}{\sim} Q\}$ and prove that it is a network bisimulation.

We want to show that if $P\{b/a\} \xrightarrow{s} P'$ then there exist Q' and s' such that $Q\{b/a\} \xrightarrow{s'} Q'$, with $s \bowtie s'$ and $P' \ \mathbf{R}_{sub} \ Q'$.

By Lemma 46 (point 2), there exist P'' and s'' such that $P \xrightarrow{s''} P''$ with $P' =$

$P''\{b/a\}$ and $s \blacktriangleright\!\blacktriangleleft s''\{b/a\}$.

By hypothesis, $P \overset{\bowtie}{\sim} Q$, then $\exists\ s'''$, Q'' such that $Q \xrightarrow{s'''} Q''$, with $s'' \rhd\!\lhd s'''$ and $P'' \overset{\bowtie}{\sim} Q'''$.

By Lemma 46 (point 1), $Q\{b/a\} \xrightarrow{s'''\{b/a\}} Q''\{b/a\}$, and we take $s' = s'''\{b/a\}$ and $Q' = Q''\{b/a\}$. Now we are done, since $P' = P''\{b/a\}\ \mathbf{R}_{sub}\ Q''\{b/a\} = Q'$ and, by Lemma 44(ii), $s \blacktriangleright\!\blacktriangleleft s''\{b/a\} \rhd\!\lhd s'''\{b/a\} = s'$. □

Example 48. It is illustrative to revisit the classical CCS counterexample that shows that strong bisimilarity is not a congruence w.r.t. substitution, already mentioned in Section 2. The translations of the two CCS processes $a.\mathbf{0} \mid \bar{b}.\mathbf{0} \sim a.\bar{b}.\mathbf{0} + \bar{b}.a.\mathbf{0}$ in CNA are respectively ${}^{\tau}\backslash_{a}.\mathbf{0} \mid {}^{b}\backslash_{\tau}.\mathbf{0}$ and ${}^{\tau}\backslash_{a}.{}^{b}\backslash_{\tau}.\mathbf{0} + {}^{b}\backslash_{\tau}.{}^{\tau}\backslash_{a}.\mathbf{0}$. But now the transition ${}^{\tau}\backslash_{a}.\mathbf{0} \mid {}^{b}\backslash_{\tau}.\mathbf{0} \xrightarrow{{}^{\tau}\backslash_{a}^{\square}\backslash_{\square}^{b}\backslash_{\tau}} \mathbf{0} \mid \mathbf{0}$ cannot be simulated by ${}^{\tau}\backslash_{a}.{}^{b}\backslash_{\tau}.\mathbf{0} + {}^{b}\backslash_{\tau}.{}^{\tau}\backslash_{a}.\mathbf{0}$ and the two processes are not network bisimilar.

5.2. Composite and Dynamic Routing in CNA

By using CNA as a modelling framework, we can now revisit the example of composite routing and prove some interesting properties. We start by introducing the simplest possible algebra for building complex routing infrastructures starting from basic building blocks. As done in Section 2, we can imagine a routing infrastructure as a box with a left and right interface and with some connections from (some of) the left channels to (some of) the right channels.

Definition 49 (Basic infrastructure). Let $\widetilde{a} = a_1, ..., a_n$ and $\widetilde{b} = b_1, ..., b_m$ be two lists of channels. A *basic routing infrastructure* R from $\widetilde{a}$ to $\widetilde{b}$, written $R(\widetilde{a}, \widetilde{b})$ is a CNA process of the form

$$R(\widetilde{a}, \widetilde{b}) \triangleq \ell_1.R(\widetilde{a}, \widetilde{b}) + ... + \ell_k.R(\widetilde{a}, \widetilde{b})$$

with $\ell_h = {}^{a_{i_h}}\backslash_{b_{j_h}}$ where $i_h \in [1, n]$ and $j_h \in [1, m]$ for any $h \in [1, k]$.

Definition 50 (Composite infrastructure). A *composite infrastructure* $R(\widetilde{a}, \widetilde{b})$ is either a basic infrastructure or the composition

$$R(\widetilde{a}, \widetilde{b}) = (\nu\, \widetilde{c})(Q(\widetilde{a}, \widetilde{c}) \mid S(\widetilde{c}, \widetilde{b}))$$

of two (composite) infrastructures $Q(\widetilde{a},\widetilde{c})$ and $S(\widetilde{c},\widetilde{b})$.

To each (composite) infrastructure $R(\widetilde{a},\widetilde{b})$ we can associate a graph $\mathcal{G}(R(\widetilde{a},\widetilde{b}))$, whose nodes are the channels appearing in the definition[6] of the process $R(\widetilde{a},\widetilde{b})$, and whose arcs are induced by the links appearing in the process, i.e. there is an arc $x \to y$ if the link ${}^{x}\backslash_{y}$ appears as a prefix in the definition of $R(\widetilde{a},\widetilde{b})$. We then have the following characterisation of the transitions admitted by $R(\widetilde{a},\widetilde{b})$.

We recall that with $||s||$ we denote the *size* of s, i.e. the number of solid links in the link chain s. Note that size is preserved by the equivalence $\blacktriangleright\!\blacktriangleleft$, but not by $\rhd\!\lhd$.

Lemma 51. *Let $R(\widetilde{a},\widetilde{b})$ be a composite infrastructure.*

1. *If $R(\widetilde{a},\widetilde{b}) \xrightarrow{s} R'$ then $R' = R(\widetilde{a},\widetilde{b})$ and there exist two nodes $a_i \in \widetilde{a}$ and $b_j \in \widetilde{b}$ of the graph $\mathcal{G}(R(\widetilde{a},\widetilde{b}))$ such that $s \bowtie {}^{a_i}\backslash_{b_j}$ and there is a path from a_i to b_j whose length is $||s||$ in the graph $\mathcal{G}(R(\widetilde{a},\widetilde{b}))$.*
2. *If there is a path of length n from a_i to b_j in the graph $\mathcal{G}(R(\widetilde{a},\widetilde{b}))$ then $R(\widetilde{a},\widetilde{b}) \xrightarrow{s} R(\widetilde{a},\widetilde{b})$ with $s \bowtie {}^{a_i}\backslash_{b_j}$ and $||s|| = n$.*

From the previous lemma, it follows that any composite infrastructure $R(\widetilde{a},\widetilde{b})$ is network bisimilar to a basic infrastructure that has one link for each possible path in $\mathcal{G}(R(\widetilde{a},\widetilde{b}))$ from one of the a_is to one of the b_js.

Definition 52. Let $R(\widetilde{a},\widetilde{b})$ be a composite infrastructure and $\mathcal{G} = \mathcal{G}(R(\widetilde{a},\widetilde{b}))$ be its corresponding graph. We denote with $P_{\mathcal{G}}(\widetilde{a},\widetilde{b})$ the basic infrastructure that offers one link for any path in $\mathcal{G}$, i.e.

$$P_{\mathcal{G}}(\widetilde{a},\widetilde{b}) \triangleq \sum_{a_i \to^* b_j \in \mathcal{G}} {}^{a_i}\backslash_{b_j}.P_{\mathcal{G}}(\widetilde{a},\widetilde{b}).$$

where $a \to^* b$ denotes the presence of a path from a to b.

Corollary 53. *Any composite infrastructure $R(\widetilde{a},\widetilde{b})$ is network bisimilar to the basic infrastructure $P_{\mathcal{G}(R(\widetilde{a},\widetilde{b}))}(\widetilde{a},\widetilde{b})$.*

[6] Without loss of generality, we can exploit alpha-conversion to assume that all restricted channels are named in a different way.

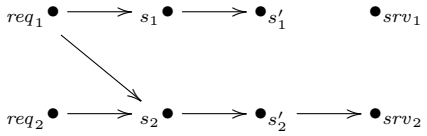
Example 54. Let us define the following basic infrastructures

$$\begin{aligned} R'(\widetilde{req},\widetilde{s}) &\triangleq {}^{req_1}\backslash_{s_1}.R'(\widetilde{req},\widetilde{s}) + {}^{req_1}\backslash_{s_2}.R'(\widetilde{req},\widetilde{s}) + {}^{req_2}\backslash_{s_2}.R'(\widetilde{req},\widetilde{s}) \\ R''(\widetilde{s},\widetilde{s'}) &\triangleq {}^{s_1}\backslash_{s'_1}.R''(\widetilde{s},\widetilde{s'}) + {}^{s_2}\backslash_{s'_2}.R''(\widetilde{s},\widetilde{s'}) \\ R'''(\widetilde{s'},\widetilde{srv}) &\triangleq {}^{s'_2}\backslash_{srv_2}.R'''(\widetilde{s'},\widetilde{srv}) \end{aligned}$$

and combine them to form the composite infrastructures

$$\begin{aligned} Q(\widetilde{req},\widetilde{s'}) &\triangleq (\nu\,\widetilde{s})(R'(\widetilde{req},\widetilde{s})|R''(\widetilde{s},\widetilde{s'})) \\ R = R(\widetilde{req},\widetilde{srv}) &\triangleq (\nu\,s')(Q(\widetilde{req},\widetilde{s'})|R'''(\widetilde{s'},\widetilde{srv})) \end{aligned}$$

Assuming all of the tuples $\widetilde{req}$, $\widetilde{s}$, $\widetilde{s'}$ and $\widetilde{srv}$ have length 2, the graph $\mathcal{G}(R)$ is depicted below



Then, it is immediately evident that the admissible transitions for R are of the form

$$R \xrightarrow{{}^{req_1}\backslash^{\tau}_{\tau}\backslash^{\tau}_{\tau}\backslash_{srv_2}} R \qquad R \xrightarrow{{}^{req_2}\backslash^{\tau}_{\tau}\backslash^{\tau}_{\tau}\backslash_{srv_2}} R$$

where, of course, many additional virtual links can be appended to the extremes of the labels (remember the Accordion Lemma 25). Consequently, R is network bisimilar to the basic infrastructure $S(\widetilde{req},\widetilde{srv}) \triangleq {}^{req_1}\backslash_{srv_2}.S(\widetilde{req},\widetilde{srv}) + {}^{req_2}\backslash_{srv_2}.S(\widetilde{req},\widetilde{srv})$.

Finally, we show that CNA is particularly convenient to model programmable infrastructures, where links can be dynamically added and removed.

Given $\widetilde{a} = a_1, ..., a_n$ and $b = b_1, ..., b_m$ let us consider the processes

$$\begin{aligned} \widehat{R}_{i,j} &\triangleq {}^{add_{i,j}}\backslash_{\tau}.R_{i,j} \\ R_{i,j} &\triangleq {}^{a_i}\backslash_{b_j}.R_{i,j} \;+\; {}^{rem_{i,j}}\backslash_{\tau}.\widehat{R}_{i,j} \;+\; {}^{add_{i,j}}\backslash_{\tau}.(R_{i,j}|R_{i,j}) \end{aligned}$$

and their parallel composition

$$R = \prod_{i=1}^{n}\prod_{j=1}^{m}\widehat{R}_{i,j}$$

where we use the shorthand $\prod_{i=1}^{n} P_i$ for the parallel composition $P_1 \mid \cdots \mid P_n$.

The idea is that an interaction involving the link ${}^{add_{i,j}}\backslash_{\tau}$ allows us to add one link from a_i to b_j and that an interaction involving the link ${}^{rem_{i,j}}\backslash_{\tau}$ allows us to remove one such link. Several links between a_i and b_j can be available at the same time, but no such link can be removed if it is not present. Initially, the process R makes no link available.

We believe that modelling infrastructures at this level of abstraction drastically improves the situation w.r.t other process algebras based on dyadic interaction, such as CCS. In fact imagine the situation where a composite programmable infrastructure is modelled in CCS: it can happen that a transfer of information is started along a viable path, but during the chain of interactions one or more of the hops are removed. As a consequence it is then impossible to deliver the request as well as acknowledge the failure. The CNA middleware guarantees that none of these troublesome scenarios can arise in the model.

5.3. Alternative Definitions

The theory of CNA is quite strong and stable and it can be extended in several directions without much efforts. Here we briefly discuss only three noteworthy possible variations of the presented framework.

Chain as prefixes. In the first variation, we could extend the syntax of CNA to allow essential chains instead of solid links as prefixes, i.e. the grammar production $P ::= \ell.P$ can be replaced by $P ::= s.P$ with s essential. This change can increase the usability of the process algebra in modelling different scenarios. For example, we can write a process such as ${}^{\tau}\backslash^{\Box}_{a}\backslash^{b}_{\Box}\backslash^{\Box}_{b}\backslash^{c}_{\Box}\backslash_{\tau}.P$ that requires an interaction from a to c via b. All of the results presented in the paper would carry over such an extension. Remarkably, network bisimilarity would still be a congruence.

Bisimilarity $\bowtie\!\sim$. In the second variation, in the bisimilarity game, we could decide to take into account the number of traversed (solid) links so to get a finer equivalence. This amounts to changing the definition of bisimulation by

requiring that the matching label s' is related to s by $\blacktriangleright\!\blacktriangleleft$ instead of $\rhd\!\lhd$. We can denote the corresponding bisimilarity as $\overset{\blacktriangleright\!\blacktriangleleft}{\sim}$. Since $\blacktriangleright\!\blacktriangleleft \subseteq \rhd\!\lhd$, it follows that $\overset{\blacktriangleright\!\blacktriangleleft}{\sim}$ is finer than $\overset{\rhd\!\lhd}{\sim}$, i.e. , it distinguishes more processes. However, as in the previous case, all the results presented in the paper would carry over this change.

Ordinary bisimilarity $\sim\!=\!\overset{\blacktriangleright\!\blacktriangleleft}{\sim}$. In the third and last variation, we could take ordinary strong bisimilarity $\sim$, by requiring exact matching of labels. Then, because of the Accordion Lemma 25, the resulting equivalence would coincide with the equivalence $\overset{\blacktriangleright\!\blacktriangleleft}{\sim}$ from the second point.

6. Concluding Remarks and Related Works

In this paper we have presented CNA as a generalisation of traditional dyadic process calculi able to deal with open multiparty interactions. These more complex forms of interactions can be represented in CNA without complicating the underlying synchronisation algebra, still quite simple and with rules similar to the ones of CCS. We have provided the calculus with an abstract semantics, called network bisimilarity that, as the strong bisimilarity of CCS, is a congruence w.r.t. all the operators of CNA. In addition, network bisimilarity is also a congruence w.r.t. substitutions. Furthermore, the theory of CNA is quite stable under several variations, such as allowing (essential) link chains as prefixes or changing the notion of network bisimulation to get finer equivalences.

Formally capturing new patterns of communication seems crucial to understand today's Internet infrastructures and their intrinsic dynamic nature. From this point of view, we have showed that CNA is particularly convenient for modelling programmable infrastructures, where links can be added and removed in a dynamic way.

Concerning the taxonomy for multiparty languages proposed in [13], we can say that CNA is *variable*, i.e. the number of participants is not fixed a priori, and *asynchronous*, i.e. not all the processes in the systems are required to make a move at each step. In contrast, CNA adopts a multi-channel mechanism which

does not work as a *gate* forcing all the involved processes to take part in the interaction. We can say that our *interaction command*, i.e. the command used to establish a multiparty interaction, only allows a multiparty interaction to happen. Thus, following this taxonomy we can classify CNA neither as *conjunctive* nor as *disjunctive* calculus.

As stated in Section 5.3, we intend to take CNA as a starting point for investigating more general forms of interaction and more advanced forms of equivalence. Several interesting directions are possible.

Some alternatives to network bisimilarity have been discussed in Section 5.3. Weak variants of them can be readily defined by considering solid link chains as internal (silent) actions (as they represent completed interactions). As usual, the corresponding equivalences will not be congruences w.r.t. choice. However, we think that the multi-party interaction available in CNA offers already a more abstract mechanism than dyadic communication, so that weak equivalences are not needed for many applications.

Another possibility, frequently used in process calculi literature, is to define the operational semantics in terms of reductions and then derive (context-closed) observational equivalences on the basis of some well-chosen observables. While the obvious choice for the observables would be link prefixes, it is difficult to set up the same methodology for CNA because open multi-party interactions can involve an unbounded number of participants and are difficult to model as reduction rules. Nevertheless, this would be an interesting research direction to explore in the future.

The name handling variant of CNA, called *link-calculus*, has been already considered in [5] and exploited in [9] to model biological interactions.

Due to space limitation, we decided to focus here on presenting the communication layer in full details and devote a companion paper to the name handling extension, which is currently in preparation. In particular, on the more applicative side, we think that the generalisation of link prefixes to *link-chain* prefixes can be very useful to encode some simple patterns of interaction directly in the action prefixes, thus enhancing the modelling power.

We plan to extend the theory to take into account some weights associated with each link, along the lines of [14]. For example, if weights are seen as costs, then processes can be compared on the basis of the overall cost of an interaction they offer, and the abstract equivalence can be refined to a preorder to reflect the fact that when two processes offer the same interactions, one is cheaper than the other. If costs are replaced with some logical information, e.g. representing the knowledge associated with the link, then an interaction can be paired with deduction and thus compared with others on the basis of the amount of information it provides. Other quantitative extensions could exploit probabilities and stochastic rates.

Another direction for future work is concerned with the cross-fertilisation between computational sciences and biology. In [9] we have shown that membrane interactions are intrinsically multi-party, by providing a faithful encoding of Brane calculi [10] in terms of *link-calculus*. Brane calculi are compartment-based calculi, introduced to model the behaviour of nested membranes in complex biological systems. We plan to include causality in the picture, so to study dependencies among interactions and track down sources of unwanted behaviours and consequences of biological reactions. Causality enriched reaction systems have already been used to study metabolic network models [15, 16], e.g. for detecting incorrect behaviour that may depend on a particular ordering of certain events, sometimes difficult to predict. The idea is to define a causal semantics for CNA and exploit static analysis techniques for approximating the causal relationships among the interactions performed by a complex system, along the lines of the abstract causal semantics proposed in [17, 18] for the Brane calculus and of the context dependent analysis presented in [19] for BioAmbients [20], another calculus for describing biological systems.

A further extension of our approach consists in the possibility of expressing non linear communication patterns in the prefixes, as allowing links of arity greater than 2 and combine them in trees, matrices or graphs.

Related Work. Among the recently presented network-aware extensions of classical calculi such as [21] (to handle explicit distribution, remote operations and process mobility), and [22] (to deal with permanent nodes crashing and links breaking), the closest proposal to ours is in [23], an extension of π-calculus, where links are named and are distinct from usual input/output actions, and there is one sender and one receiver (the output includes the final receiver name). In the name-passing variant of CNA [5], links can carry message tuples, and each participant can play both the sender and the receiver rôle. This extended semantics recalls the concurrent semantics in [23], where concurrent transmissions can be observed in the form of a multi-set of routing paths. In our case the collected links are organised in a link chain.

In [24], the authors present a general framework to extend synchronisation algebras [25] with name mobility that could be easily adapted to many other high-level kinds of synchronisation, like the one we need, but with a more complex machinery. More sophisticated forms of synchronisations, with a fixed number of processes, are introduced in π-calculus in [26] (joint input) and in [27] (polyadic synchronisation). The focus of [28] is instead on the expressiveness of an asynchronous CCS equipped with joint inputs allowing the interactions of n processes, proving that there is no truly distributed implementation of operators synchronising more than three processes. As in the Join-calculus [29], and differently from our approach, participants can act either as senders or as receivers.

In [30], a conservative extension of CCS, called A^2CCS, is studied that is able to model multi-party synchronisation. The mechanism is realised as an atomic sequence of dyadic synchronisations of arbitrary lengths, but imposes some constraints that make the parallel operator non associative and therefore more difficult to use as a model.

Finally, in [31], a distributed version of the π-calculus for handling names, considered as localised to their owners, in concurrent and distributed systems made of mobile processes. Each process is indeed equipped with a local name environment. When a name is exported, it is equipped with the information

needed to point back to its local environment, thus keeping track of the origin of mobile agents in multi-hop travel on the network. Communications are not open, but are instead controlled by a distributed name manager that keeps distinct the names generated by different environments.

As a last remark, it is worth noting that the operational semantics of CNA allows a link prefix to participate in infinitely many transitions that account for the presence of the link within chains of any length. Thus, a direct implementation of the CNA semantics that can be used for verification is not immediate. A possible solution to overcome this problem is the definition of a symbolic semantics [32] that collapses in a single transition all the transitions labelled with link chains composed with the same set of solid links.

References

[1] D. Kreutz, F. M. V. Ramos, P. J. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proceedings of the IEEE 103 (1) (2015) 14–76.

[2] K. Honda, N. Yoshida, M. Carbone, Multiparty asynchronous session types, in: G. C. Necula, P. Wadler (Eds.), Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, ACM, 2008, pp. 273–284.

[3] H. Hüttel, I. Lanese, V. T. Vasconcelos, L. Caires, M. Carbone, P. Deniélou, D. Mostrous, L. Padovani, A. Ravara, E. Tuosto, H. T. Vieira, G. Zavattaro, Foundations of session types and behavioural contracts, ACM Comput. Surv. 49 (1) (2016) 3.

[4] R. Milner, Communication and concurrency, PHI Series in computer science, Prentice Hall, 1989.

[5] C. Bodei, L. Brodo, R. Bruni, Open multiparty interaction, in: Recent Trends in Algebraic Development Techniques, 21st International Workshop,

WADT 2012, Vol. 7841 of Lecture Notes in Computer Science, Springer, 2012, pp. 1–23.

[6] R. Milner, Communicating and mobile systems: the pi-calculus, Cambridge University Press, 1999.

[7] A. Gordon, L. Cardelli, Equational properties of mobile ambients, Mathematical Structures in Computer Science 13 (3) (2003) 371–408.

[8] L. Brodo, On the expressiveness of pi-calculus for encoding mobile ambients, Mathematical Structures in Computer Science (2016) 1–39.

[9] C. Bodei, L. Brodo, R. Bruni, D. Chiarugi, A flat process calculus for nested membrane interactions, Sci. Ann. Comp. Sci. 24 (1) (2014) 91–136.

[10] L. Cardelli, Brane calculi, in: Proc. of Computational Methods in Systems Biology (CMSB'04), Vol. 3082 of Lecture Notes in Computer Science, Springer, 2005, pp. 257–280.

[11] G. Winskel, Synchronization trees, Theoretical Computer Science 34 (1984) 33–82.

[12] C. A. R. Hoare, Communicating Sequential Processes, Prentice-Hall, 1985.

[13] Y.-J. Joung, S. Smolka, A comprehensive study of the complexity of multiparty interaction, J. ACM 43 (1) (1996) 75–115.

[14] M. Hennessy, A calculus for costed computations, Logical Methods in Computer Science 7 (1).

[15] C. Bodei, A. Bracciali, D. Chiarugi, On deducing causality in metabolic networks, BMC Bioinformatics 9 (S-4).

[16] R. Barbuti, R. Gori, F. Levi, P. Milazzo, Investigating dynamic causalities in reaction systems, Theor. Comput. Sci. 623 (2016) 114–145.

[17] C. Bodei, R. Gori, F. Levi, An analysis for causal properties of membrane interactions, Electr. Notes Theor. Comput. Sci. 299 (2013) 15–31.

[18] C. Bodei, R. Gori, F. Levi, Causal static analysis for brane calculi, Theor. Comput. Sci. 587 (2015) 73–103.

[19] H. Pilegaard, H. R. Nielson, F. Nielson, Context dependent analysis of bioambients, in: Simulation and Verification of Dynamic Systems, Vol. 06161 of Dagstuhl Seminar Proceedings, 2006.

[20] A. Regev, E. Panina, W. Silverman, L. Cardelli, E. Shapiro, Bioambients: An abstraction for biological compartements, Theor. Comput. Sci. 325 (1) (2004) 141–167.

[21] A. Francalanza, M. Hennessy, A theory of system behaviour in the presence of node and link failure, Information and Computation 206 (6) (2008) 711–759.

[22] R. D. Nicola, D. Gorla, R. Pugliese, Basic observables for a calculus for global computing, Information and Computation 205 (10) (2007) 1491–1525.

[23] U. Montanari, M. Sammartino, Network conscious pi-calculus: a concurrent semantics, in: Proc. of Mathematical Foundations of Programming Semantics (MFPS 2012), Electronic Notes in Theoretical Computer Science 286, Elsevier, 2012, pp. 291–306.

[24] R. Bruni, I. Lanese, Parametric synchronizations in mobile nominal calculi, Theoretical Computer Science 402 (2-3) (2008) 102–119.

[25] G. Winskel, Synchronization trees, Theoretical Computer Science 34 (1-2) (1984) 33–82.

[26] U. Nestmann, On the expressive power of joint input, Electronic Notes in Theoretical Computer Science 16 (2).

[27] M. Carbone, S. Maffeis, On the expressive power of polyadic synchronisation in pi-calculus, Nordic Journal of Computing 10 (2) (2003) 70–98.

[28] C. Laneve, A. Vitale, The expressive power of synchronizations, in: Proc. of Logic in Computer Science (LICS 2010), IEEE Computer Society, 2010, pp. 382–391.

[29] C. Fournet, G. Gonthier, The reflexive CHAM and the Join-calculus, in: Proc. of ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL 1996), ACM Press, 1996, pp. 372–385.

[30] R. Gorrieri, C. Versari, An operational petri net semantics for A^2CCS, Fundamenta Informaticae 109 (2) (2011) 135–160.

[31] C. Bodei, P. Degano, C. Priami, Names of the pi-calculus agents handled locally, Theoretical Computer Science 253 (2) (2001) 155–184.

[32] L. Brodo, C. Olarte, Symbolic semantics for multiparty interactions in the link-calculus, in: Proc. of SOFSEM 2017: Theory and Practice of Computer Science - 43rd International Conference on Current Trends in Theory and Practice of Computer Science, Vol. 10139 of Lecture Notes in Computer Science, Springer, 2017, pp. 62–75.

Appendix A. Proofs of Technical Results

In this appendix we restate the lemmata presented earlier in the paper and gives the proofs of their correctness.

Appendix A.1. Proofs of Section 3

We recall Lemma 11 (the original appears on p. 16).

Lemma 11. *For any ℓ, ℓ', s, s':*

(i) The merge of links and link chains is commutative and associative.

(ii) $\ell \bullet \ell' = {}^{\square}\backslash_{\square}$ if and only if $\ell = \ell' = {}^{\square}\backslash_{\square}$.

(iii) If s is solid, then for any s' we have $s \bullet s' = \bot$.

Proof. We prove the three items separately.

(i) Trivial, by the fact that the underlying operation on actions $\alpha \bullet \beta$ is commutative and associative.

(ii) The thesis follows by applying the definition of merge. Let $\ell = {}^{\alpha}\backslash_{\beta}$ and $\ell' = {}^{\alpha'}\backslash_{\beta'}$ then $\ell \bullet \ell' = {}^{(\alpha\bullet\alpha')}\backslash_{(\beta\bullet\beta')}$, where $(\alpha \bullet \alpha') = \Box$ iff $\alpha = \alpha' = \Box$. Similarly, $(\beta \bullet \beta') = \Box$ iff $\beta = \beta' = \Box$.

(iii) Since s is solid, its length $n = |s|$ is greater than zero. If $|s'| \neq n$ then $s \bullet s' = \bot$. Otherwise, let $s = \ell_1 ... \ell_n$ and $s' = \ell'_1 ... \ell'_n$. Since link chains cannot be made of virtual links only, there is at least a position i in s' such that ℓ'_i is solid. Then, $\ell_i \bullet \ell'_i = \bot$ because, since s is solid, also ℓ_i is solid. As a consequence, $s \bullet s' = \bot$.

□

We recall Lemma 12 (the original appears on p. 17).

Lemma 12. *Let s, s', and s'' be three link chains such that $(s' \bullet s'') \bowtie s$, then there must exist s_1 and s_2 such that $s_1 \bowtie s'$ and $s_2 \bowtie s''$ with $s_1 \bullet s_2 = s$.*

Proof. We prove that the axioms in Definition 8, when applied in either direction, satisfy the property. Then, by transitivity, the thesis holds for all the elements in each equivalent class of $\bowtie$. The proof proceeds by cases on the axioms of $\bowtie$.

case $[s_0 \bowtie {s_0}^{\Box}\backslash_{\Box}]$ Let $s' \bullet s'' = s_0$ and $s = {s_0}^{\Box}\backslash_{\Box}$. Now we have to find $s_1 \bowtie s'$ and $s_2 \bowtie s''$ such that $s_1 \bullet s_2 = s$. To this aim, we set[7] $s_1 = {s'}^{\Box}\backslash_{\Box}$ and $s_2 = {s''}^{\Box}\backslash_{\Box}$. By definition of the merge operator, $\bullet$, we get that $s_1 \bullet s_2 = {s'}^{\Box}\backslash_{\Box} \bullet {s''}^{\Box}\backslash_{\Box} = (s' \bullet s'')^{\Box}\backslash_{\Box} = {s_0}^{\Box}\backslash_{\Box} = s$.

case $[{s_0}^{\Box}\backslash_{\Box} \bowtie s_0]$ By hypothesis, $s' \bullet s'' = {s_0}^{\Box}\backslash_{\Box}$ and $s = s_0$. By definition of the merge operator, $\bullet$, we get $s' = {s_1}^{\Box}\backslash_{\Box}$ and $s'' = {s_2}^{\Box}\backslash_{\Box}$, for suitable s_1 and s_2. Then, we have $s = s_1 \bullet s_2$.

[7] Note that s_1 and s_2 are link chains since otherwise $(s' \bullet s'')^{\Box}\backslash_{\Box} = s$ would not be defined.

case $[s_0{}^{\square}\backslash^{\square}_{\square}\backslash_{\square}s'_0 \bowtie s_0{}^{\square}\backslash_{\square}s'_0]$ Let $s' \bullet s'' = s_0{}^{\square}\backslash^{\square}_{\square}\backslash_{\square}s'_0$ and $s = s_0{}^{\square}\backslash_{\square}s'_0$. Therefore it must be the case that $s' = s'_1{}^{\square}\backslash^{\square}_{\square}\backslash_{\square}s''_1$ and $s'' = s'_2{}^{\square}\backslash^{\square}_{\square}\backslash_{\square}s''_2$ with $s'_1 \bullet s'_2 = s_0$ and $s''_1 \bullet s''_2 = s'_0$. Then we let $s_1 = s'_1{}^{\square}\backslash_{\square}s''_1$ and $s_2 = s'_2{}^{\square}\backslash_{\square}s''_2$ and we get $s_1 \bullet s_2 = (s'_1 \bullet s'_2)^{\square}\backslash_{\square}(s''_1 \bullet s''_2) = s_0{}^{\square}\backslash_{\square}s'_0 = s$.

case $[s_0{}^{\square}\backslash_{\square}s'_0 \bowtie s_0{}^{\square}\backslash^{\square}_{\square}\backslash_{\square}s'_0]$ Let $s' \bullet s'' = s_0{}^{\square}\backslash_{\square}s'_0$ and $s = s_0{}^{\square}\backslash^{\square}_{\square}\backslash_{\square}s'_0$. Therefore it must be the case that $s' = s'_1{}^{\square}\backslash_{\square}s''_1$ and $s'' = s'_2{}^{\square}\backslash_{\square}s''_2$ with $s'_1 \bullet s'_2 = s_0$ and $s''_1 \bullet s''_2 = s'_0$. Then we let $s_1 = s'_1{}^{\square}\backslash^{\square}_{\square}\backslash_{\square}s''_1$ and $s_2 = s'_2{}^{\square}\backslash^{\square}_{\square}\backslash_{\square}s''_2$ and we get $s_1 \bullet s_2 = (s'_1 \bullet s'_2)^{\square}\backslash^{\square}_{\square}\backslash_{\square}(s''_1 \bullet s''_2) = s_0{}^{\square}\backslash^{\square}_{\square}\backslash_{\square}s'_0 = s$.

case $[s_0{}^{\alpha}\backslash^{\square}_{a}\backslash^{a}_{\square}\backslash_{\beta}s'_0 \bowtie s_0{}^{\alpha}\backslash^{a}_{a}\backslash_{\beta}s'_0]$ Let $s' \bullet s'' = s_0{}^{\alpha}\backslash^{\square}_{a}\backslash^{a}_{\square}\backslash_{\beta}s'_0$ and $s = s_0{}^{\alpha}\backslash^{a}_{a}\backslash_{\beta}s'_0$. Then, there are four possible cases:

$s' = s'_1{}^{\square}\backslash^{\square}_{\square}\backslash^{\square}_{\square}\backslash_{\square}s''_1$ and $s'' = s'_2{}^{\alpha}\backslash^{\square}_{a}\backslash^{a}_{\square}\backslash_{\beta}s''_2$

$s' = s'_1{}^{\alpha}\backslash^{\square}_{a}\backslash^{\square}_{\square}\backslash_{\square}s''_1$ and $s'' = s'_2{}^{\square}\backslash^{\square}_{\square}\backslash^{a}_{\square}\backslash_{\beta}s''_2$

$s' = s'_1{}^{\square}\backslash^{\square}_{\square}\backslash^{a}_{\square}\backslash_{\beta}s''_1$ and $s'' = s'_2{}^{\alpha}\backslash^{\square}_{a}\backslash^{\square}_{\square}\backslash_{\square}s''_2$

$s' = s'_1{}^{\alpha}\backslash^{\square}_{a}\backslash^{a}_{\square}\backslash_{\beta}s''_1$ and $s'' = s'_2{}^{\square}\backslash^{\square}_{\square}\backslash^{\square}_{\square}\backslash_{\square}s''_2$

with $s'_1 \bullet s'_2 = s_0$ and $s''_1 \bullet s''_2 = s'_0$.

We only show the first case, as the other ones are similar.

Now we let $s_1 = s'_1{}^{\square}\backslash^{\square}_{\square}\backslash_{\square}s''_1 \bowtie s'$ and $s_2 = s'_2{}^{\alpha}\backslash^{a}_{a}\backslash_{\beta}s'_2 = s$ and we are done since $s_1 \bullet s_2 = (s_1 \bullet s'_2)^{\alpha}\backslash^{a}_{a}\backslash_{\beta}(s''_1 \bullet s''_2)$.

The remaining cases, i.e. $s_0 \bowtie {}^{\square}\backslash_{\square}s_0$, ${}^{\square}\backslash_{\square}s_0 \bowtie s_0$ and $s_0{}^{\alpha}\backslash^{a}_{a}\backslash_{\beta}s'_0 \bowtie s_0{}^{\alpha}\backslash^{\square}_{a}\backslash^{a}_{\square}\backslash_{\beta}s'_0$, have similar proofs and are omitted. □

We recall Lemma 15 (the original appears on p. 18).

Lemma 15. *For any a, b, ℓ, s, s'*

(i) $(\nu\, a)\ell = {}^{\square}\backslash_{\square}$ *if and only if* $\ell = {}^{\square}\backslash_{\square}$.

(ii) $(\nu\, a)(s \bullet s') = s \bullet (\nu\, a)s'$ *if a does not occur in s.*

(iii) $(\nu\, a)(\nu\, b)s = (\nu\, b)(\nu\, a)s$.

Proof. The proof derives from the definitions of the restriction $(\nu\,)$ operator and of the merge operator $\bullet$, both defined on link chains.

(i) Obvious, as $(\nu\, a)\alpha \neq \Box$ if $\alpha \neq \Box$.

(ii) If $(\nu\, a)s' = \bot$ it means that a is unmatched in s' and since a does not appear in s it remains unmatched in $s \bullet s'$. Otherwise, a is matched in s' and since solid links are preserved by $\bullet$ it remains matched in $s \bullet s'$, then renaming a to τ before or after the merge does not change the result.

(iii) Obvious as $(\nu\, a)(\nu\, b)\alpha = (\nu\, b)(\nu\, a)\alpha$ for any α.

□

We recall Lemma 17 (the original appears on p. 18).

Lemma 17. *Let s and s' be two link chains such that $(\nu\, a)s$ is defined and $(\nu\, a)s \bowtie s'$, then there exists s'' such that $s' = (\nu\, a)s''$ and $s \bowtie s''$.*

Proof. We prove that the axioms, when applied in either direction, satisfy the property. Then, by transitivity, the thesis holds for all the elements in each equivalent class of $\bowtie$. The proof proceeds by cases on axioms of $\bowtie$.

case $[s_0 \bowtie {s_0}^{\Box}\backslash_{\Box}]$ Let $(\nu\, a)s = s_0$ and $s' = {s_0}^{\Box}\backslash_{\Box}$. Then, we set $s'' = {s}^{\Box}\backslash_{\Box}$, and it is immediate to verify that a is matched in s'', as it is in s, thus we can write $s' = (\nu\, a)s''$ (with $(\nu\, a)s \bowtie (\nu\, a)s''$) and we get that $s \bowtie s''$.

case $[{s_0}^{\Box}\backslash_{\Box} \bowtie s_0]$ Let $(\nu\, a)s = {s_0}^{\Box}\backslash_{\Box}$ and $s' = s_0$. Then it must exists s'' s.t. $s = {s''}^{\Box}\backslash_{\Box}$ with $s_0 = (\nu\, a)s''$. The thesis follows as $s' = s_0 = (\nu\, a)s''$ and clearly $s \bowtie s''$.

case $[{s_0}^{\Box}\backslash_{\Box} s_0' \bowtie {s_0}^{\Box}\backslash^{\Box}_{\Box}\backslash_{\Box} s_0']$ Let $(\nu\, a)s = {s_0}^{\Box}\backslash_{\Box} s_0'$ and $s' = {s_0}^{\Box}\backslash^{\Box}_{\Box}\backslash_{\Box} s_0'$. Then it must be $s = {s_1}^{\Box}\backslash_{\Box} s_2$ for some s_1 and s_2 with $(\nu\, a)s_1 = s_0$ and $(\nu\, a)s_2 = s_0'$. We set $s'' = {s_1}^{\Box}\backslash^{\Box}_{\Box}\backslash_{\Box} s_2$, from which the thesis immediately follows.

case $[{s_0}^{\alpha}\backslash^{\Box}_{b}\backslash^{b}_{\Box}\backslash_{\beta} s_0' \bowtie {s_0}^{\alpha}\backslash^{b}_{b}\backslash_{\beta} s_0']$ Let $(\nu\, a)s = {s_0}^{\alpha}\backslash^{\Box}_{b}\backslash^{b}_{\Box}\backslash_{\beta} s_0'$ and $s' = {s_0}^{\alpha}\backslash^{b}_{b}\backslash_{\beta} s_0'$. As a cannot appear in $(\nu\, a)s$, it must be the case that $a \neq b$ and $s = {s_1}^{\Box}\backslash_{\Box} s_2$ for some s_1 and s_2 with $(\nu\, a)s_1 = {s_0}^{\alpha}\backslash_{b}$ and $(\nu\, a)s_2 = {}^{b}\backslash_{\beta} s_0'$. We set $s'' = s_1 s_2$, from which the thesis immediately follows.

case $[{s_0}^{\alpha}\backslash_{b}^{b}\backslash_{\beta} s_0' \bowtie {s_0}^{\alpha}\backslash_{b}^{\square}\backslash_{\square}^{b}\backslash_{\beta} s_0']$ Let $(\nu\, a)s = {s_0}^{\alpha}\backslash_{b}^{b}\backslash_{\beta} s_0'$ and $s' = {s_0}^{\alpha}\backslash_{b}^{\square}\backslash_{\square}^{b}\backslash_{\beta} s_0'$. As a cannot appear in $(\nu\, a)s$, it must be the case that $a \neq b$ and $s = s_1 s_2$ for some s_1 and s_2 with $(\nu\, a)s_1 = {s_0}^{\alpha}\backslash_{b}$ and $(\nu\, a)s_2 = {}^{b}\backslash_{\beta} s_0'$. We set $s'' = {s_1}^{\square}\backslash_{\square} s_2$, from which the thesis immediately follows.

We omit the remaining cases that are analogous. □

We recall Lemma 19 (the original appears on p. 18).

Lemma 19. *For any $a, \phi, \psi, \ell, s, s'$*

(i) $\ell[\phi] = {}^{\square}\backslash_{\square}$ *if and only if* $\ell = {}^{\square}\backslash_{\square}$.

(ii) $(s \bullet s')[\phi] = s[\phi] \bullet (s'[\phi])$.

(iii) $((\nu\, a)s)[\phi] = (\nu\, \phi(a))(s[\phi])$.

(iv) $s[\phi][\psi] = s[\psi \circ \phi]$.

(v) If $s \bowtie s'$ *then* $s[\phi] \bowtie s'[\phi]$.

Proof. The proof of the points (i), (ii), (iii) derives from the definitions of the renaming function ϕ, of the merge operator $\bullet$, and of restriction operator $(\nu\ )$, all defined on link chains.

(i) Obvious, as $\ell[\phi] \neq {}^{\square}\backslash_{\square}$ if $\ell \neq {}^{\square}\backslash_{\square}$.

(ii) Let $s = \ell_1 ... \ell_n$ and $s' = \ell_1' ... \ell_n'$, with $\ell_i = {}^{\alpha_i}\backslash_{\beta_i}$ and $\ell_i' = {}^{\alpha_i'}\backslash_{\beta_i'}$, for all $i \in [1, n]$. Then, by definition of merge and renaming:

$$(s \bullet s')[\phi] = ((\ell_1 \bullet \ell_1') \cdots (\ell_n \bullet \ell_n'))[\phi] = (\ell_1 \bullet \ell_1')[\phi] \cdots (\ell_n \bullet \ell_n')[\phi].$$

Since, for all $i \in [1, n]$, $(\ell_i \bullet \ell_i')[\phi] = {}^{(\phi(\alpha_i) \bullet \phi(\alpha_i'))}\backslash_{(\phi(\beta_i) \bullet \phi(\beta_i'))} = {}^{\phi(\alpha_i)}\backslash_{\phi(\beta_i)} \bullet {}^{\phi(\alpha_i')}\backslash_{\phi(\beta_i')} = (\ell_i[\phi] \bullet \ell_i'[\phi])$, we can conclude that $(s \bullet s')[\phi] = s[\phi] \bullet (s'[\phi])$.

(iii) Let $s = \ell_1 ... \ell_n$ with $\ell_i = {}^{\alpha_i}\backslash_{\beta_i}$ then $((\nu\, a)s)[\phi] = (((\nu\, a)\ell_1) \dots ((\nu\, a)\ell_n))[\phi]$, that amounts to $(((\nu\, a)\ell_1)[\phi] \dots ((\nu\, a)\ell_n)[\phi])$. Note that for any α we have $\phi((\nu\, a)\alpha) = (\nu\, \phi(a))\phi(\alpha)$. In fact, if $\alpha = a$ then $\phi((\nu\, a)a) = \phi(\tau) = \tau$ and $(\nu\, \phi(a))\phi(a) = \tau$. If instead $\alpha \neq a$, then $\phi(\alpha) \neq \phi(a)$ (because ϕ is a bijection) and thus $\phi((\nu\, a)\alpha) = \phi(\alpha) = (\nu\, \phi(a))\phi(\alpha)$. Since, for all $i \in [1, n]$,

$((\nu\, a)\ell_i))[\phi] = ((\nu\, a)^{\alpha_i}\backslash_{\beta_i})[\phi] = {}^{(\nu\, a)\alpha_i}\backslash_{(\nu\, a)\beta_i}[\phi] = {}^{\phi((\nu\, a)\alpha_i)}\backslash_{\phi((\nu\, a)\beta_i)} = {}^{(\nu\, \phi(a))\phi(\alpha_i)}\backslash_{(\nu\, \phi(a))\phi(\beta_i)} = (\nu\, \phi(a))({}^{\phi(\alpha_i)}\backslash_{\phi(\beta_i)}) = (\nu\, \phi(a))({}^{\alpha_i}\backslash_{\beta_i}[\phi])$, we can conclude that $((\nu\, a)s)[\phi] = (\nu\, \phi(a))(s[\phi])$.

(iv) The proof of (iv) derives from the compositionality of renaming functions, in particular, ${}^{\alpha}\backslash_{\beta}[\phi][\psi] = {}^{\phi(\alpha)}\backslash_{\phi(\beta)}[\psi] = {}^{\psi\circ\phi(\alpha)}\backslash_{\psi\circ\phi(\beta)} = ({}^{\alpha}\backslash_{\beta})[\psi \circ \phi]$.

(v) To prove (v), we prove that the axioms, when applied in either direction, satisfy the property. Then, by transitivity, the thesis holds for all the elements in each equivalent class of $\bowtie$. The proof proceeds by cases on axioms of $\bowtie$. For the sake of simplicity, we show only one case.

case $[s_0 \bowtie {s_0}^{\Box}\backslash_{\Box}]$ Let $s = s_0$ and $s' = {s_0}^{\Box}\backslash_{\Box}$. We have to show that $s[\phi] \bowtie s'[\phi]$ i.e. that $s_0[\phi] \bowtie {s_0}^{\Box}\backslash_{\Box}[\phi]$, where ${s_0}^{\Box}\backslash_{\Box}[\phi] = s_0[\phi]^{\Box}\backslash_{\Box}$, since ϕ distributes over the single links. Therefore, we obtain that $s_0[\phi] \bowtie s_0[\phi]^{\Box}\backslash_{\Box}$.

□

We recall Lemma 20 (the original appears on p. 19).

Lemma 20. *Let ϕ be a channel renaming function and s, s' be two link chains such that $s[\phi] \bowtie s'$, then there exists s'' such that $s' = s''[\phi]$ and $s \bowtie s''$.*

Proof. Since ϕ is a bijection, we take its inverse ϕ^{-1} and let $s'' \triangleq s'[\phi^{-1}]$. Then the thesis holds by Lemma 19(iv–v): $s' = s''[\phi]$ trivially holds, since $s''[\phi] = s'[\phi^{-1}][\phi] = s'$, and $s \bowtie s'[\phi^{-1}]$ holds because $s[\phi] \bowtie s'$ implies $s = s[\phi][\phi^{-1}] \bowtie s'[\phi^{-1}]$. □

Appendix A.2. Proofs of Section 5

We recall Lemma 33 (the original appears on p. 28).

Lemma 33. *All of the following properties hold for any link chain s.*

(i) There exists an essential link chain s' such that $s \bowtie\!\triangleleft s'$.

(ii) If s is essential, for any essential s' such that $s \bowtie\!\triangleleft s'$, then $s = s'$.

Proof.

(i) Take s and let

- n be the number of adjacent solid links in s,
- m be the number of adjacent virtual links in s,
- k be the number of virtual links at the extremes of s.

For example, for $s = {}^{\square}\backslash{}^{\square}_{\square}\backslash{}^{a}_{\square}\backslash{}^{\tau}_{\tau}\backslash{}^{b}_{b}\backslash{}^{\square}_{c}\backslash{}^{c}_{\square}\backslash{}^{\square}_{d}\backslash{}^{\square}_{\square}\backslash{}^{e}_{\square}\backslash_{\tau}$, we have $n = 3$, $m = 4$ and $k = 2$.

We prove the existence of s' by induction on $v(s) = n + m + k$.

- For the base case, if $v(s) = 0$ then s is essential and we are done.
- For the inductive case, suppose $v(s) > 0$. Then at least one of n, m, k is greater than 0.

  If $n > 0$, then there are two adjacent links in s such as ${}^{\alpha}\backslash{}^{a}_{a}\backslash_{\beta}$ or ${}^{\alpha}\backslash{}^{\tau}_{\tau}\backslash_{\beta}$. In the former case, we can apply the last axiom of $\blacktriangleright\!\blacktriangleleft$ (Definition 8) to introduce a virtual link between the matched action a and decrement by one the number of adjacent solid links. In the latter case, we can apply the rightmost axiom of $\rhd\!\lhd$ (Definition **??**) to eliminate the matched τ actions and decrement by one the number of adjacent solid links.

  If $m > 0$, then there are two adjacent virtual links in s and we can apply the top-right axiom of $\blacktriangleright\!\blacktriangleleft$ (Definition 8) to decrement by one the number of adjacent virtual links.

  If $k > 0$ we can apply one of the leftmost axioms of $\blacktriangleright\!\blacktriangleleft$ (Definition 8) to decrement by one the number of virtual links at the extremes.

  In all cases we get a chain $s'' \rhd\!\lhd s$ with $v(s'') = v(s) - 1$ and, by inductive hypothesis, there is an essential link chain s' such that $s' \rhd\!\lhd s''$. Thus, by transitivity, we have $s' \rhd\!\lhd s$.

(ii) By contradiction, let $s = \ell_1{}^{\square}\backslash_{\square}\ell_2 \ldots {}^{\square}\backslash_{\square}\ell_n$ and $s' = \ell'_1{}^{\square}\backslash_{\square}\ell'_2 \ldots {}^{\square}\backslash_{\square}\ell'_m$ be two essential link chains such that $s \rhd\!\lhd s'$ and $s \neq s'$. Without loss of

generality, assume s and s' are chosen such that the length of s is the minimal one for which such a counterexample exists. If $\ell_1 = \ell'_1$, then $\ell_2 \dots {}^{\square}\backslash_{\square}\ell_n$ and $\ell'_2 \dots {}^{\square}\backslash_{\square}\ell'_m$ would provide a shorter counterexample, contradicting the hypothesis of minimality for n. Thus it must be $\ell_1 \neq \ell'_1$. Let $\ell_1 = {}^{\alpha_1}\backslash_{\beta_1}$ and $\ell'_1 = {}^{\alpha'_1}\backslash_{\beta'_1}$. Now we can notice that the axioms for $\rtimes\ltimes$ preserves the leftmost non-virtual symbol of a link chain. Thus $\alpha_1 = \alpha'_1$, otherwise $s \rtimes\ltimes s'$ would not hold. Finally, we notice that any non-virtual symbol adjacent to a virtual link is preserved by the axioms. Thus $\beta_1 = \beta'_1$ and $\ell_1 = \ell'_1$ leading to a contradiction. □

We recall Lemma 35 (the original appears on p. 28).

Lemma 35. *If $s \rtimes\ltimes s'$, then for any a such that $(\nu\, a)s \neq \bot$ there exists $s'' \blacktriangleright\!\blacktriangleleft s'$ such that $(\nu\, a)s'' \neq \bot$ and $(\nu\, a)s \rtimes\ltimes (\nu\, a)s''$.*

Proof. We prove that the property holds for the axioms of $\rtimes\ltimes$ when applied in each direction, then the fact that the property is preserved by the rules for equivalence is immediate. We prove only some cases; the remaining ones are similar.

case [${s_1}^{\alpha}\backslash^{\tau}_{\tau}\backslash_{\beta} s_2 \rtimes\ltimes {s_1}^{\alpha}\backslash_{\beta} s_2$] Let $s = {s_1}^{\alpha}\backslash^{\tau}_{\tau}\backslash_{\beta} s_2$ and $s' = {s_1}^{\alpha}\backslash_{\beta} s_2$. Since $(\nu\, a)s \neq \bot$ it means that a is matched in s and thus it is matched in s', which differs from s only for the removal of τ, and we put $s'' = s'$. Then we have

$$\begin{aligned}(\nu\, a)s'' &= ((\nu\, a)s_1)^{(\nu\, a)\alpha}\backslash_{(\nu\, a)\beta}((\nu\, a)s_2)\\ &\rtimes\ltimes ((\nu\, a)s_1)^{(\nu\, a)\alpha}\backslash^{\tau}_{\tau}\backslash_{(\nu\, a)\beta}((\nu\, a)s_2)\\ &= (\nu\, a)s.\end{aligned}$$

case [${s_1}^{\alpha}\backslash^{a}_{a}\backslash_{\beta} s_2 \rtimes\ltimes {s_1}^{\alpha}\backslash^{\square}_{a}\backslash^{a}_{\square}\backslash_{\beta} s_2$] Let $s = {s_1}^{\alpha}\backslash^{a}_{a}\backslash_{\beta} s_2$ and $s' = {s_1}^{\alpha}\backslash^{\square}_{a}\backslash^{a}_{\square}\backslash_{\beta} s_2$. We let $s'' = s \blacktriangleright\!\blacktriangleleft s'$ and we are done, since $(\nu\, a)s'' = (\nu\, a)s$ is defined by hypothesis.

case [${s_1}^{\alpha}\backslash^{\square}_{a}\backslash^{a}_{\square}\backslash_{\beta} s_2 \rtimes\ltimes {s_1}^{\alpha}\backslash^{a}_{a}\backslash_{\beta} s_2$] Let $s = {s_1}^{\alpha}\backslash^{\square}_{a}\backslash^{a}_{\square}\backslash_{\beta} s_2$ and $s' = {s_1}^{\alpha}\backslash^{a}_{a}\backslash_{\beta} s_2$. Since $(\nu\, a)s$ is not defined, we are done.

□ □

We recall Lemma 36 (the original appears on p. 29).

Lemma 36. *If $s_1 \rtimes\ltimes s_1'$, then for any s_2 such that $s_2 \bullet s_1 \neq \bot$ there exist two link chains $s_2' \bowtie s_2$ and $s_1'' \bowtie s_1'$ such that $s_2' \bullet s_1'' \neq \bot$ and $s_2 \bullet s_1 \rtimes\ltimes s_2' \bullet s_1''$.*

Proof. We prove that the property holds for the axioms of $\rtimes\ltimes$, then the fact that the property is preserved by the rules for equivalence is immediate. We prove only two cases, the remaining ones are similar.

case $[s^{\alpha}\backslash^{\tau}_{\tau}\backslash_{\beta} s' \rtimes\ltimes s^{\alpha}\backslash_{\beta} s'\ ]$ We have $s_1 = s^{\alpha}\backslash^{\tau}_{\tau}\backslash_{\beta} s'$ and $s_1' = s^{\alpha}\backslash_{\beta} s'$. By definition of valid link, the links ${}^{\alpha}\backslash_{\tau}$ and ${}^{\tau}\backslash_{\beta}$ are solid, i.e. $\alpha \neq \Box$ and $\beta \neq \Box$. Then, since $s_2 \bullet s_1 \neq \bot$ we infer that $s_2 = s''^{\Box}\backslash^{\Box}_{\Box}\backslash_{\Box} s'''$ for some s'', s'' such that $s'' \bullet s \neq \bot$, and $s''' \bullet s' \neq \bot$. Then we take $s_2' = s''^{\Box}\backslash_{\Box} s''' \bowtie s_2$ and $s_1'' = s_1'$ and we are done, since $s_2 \bullet s_1 = (s'' \bullet s)^{\alpha}\backslash^{\tau}_{\tau}\backslash_{\beta}(s''' \bullet s') \rtimes\ltimes (s'' \bullet s)^{\alpha}\backslash_{\beta}(s''' \bullet s') = s_2' \bullet s_1''$.

case $[s^{\alpha}\backslash^{\Box}_{a}\backslash^{a}_{\Box}\backslash_{\beta} s' \rtimes\ltimes s^{\alpha}\backslash^{a}_{a}\backslash_{\beta} s'\ ]$ We have $s_1 = s^{\alpha}\backslash^{\Box}_{a}\backslash^{a}_{\Box}\backslash_{\beta} s'$ and $s_1' = s^{\alpha}\backslash^{a}_{a}\backslash_{\beta} s'$. Then, there are two possibilities: either (a) $s_2 = s''^{\Box}\backslash^{\Box}_{\Box}\backslash^{\Box}_{\Box}\backslash_{\Box} s'''$, or (b) $s_2 = s''^{\Box}\backslash^{a}_{\Box}\backslash^{\Box}_{a}\backslash_{\Box} s'''$ with $s'' \bullet s \neq \bot$, and $s''' \bullet s' \neq \bot$. In the case (a), we take $s_2' = s''^{\Box}\backslash^{\Box}_{\Box}\backslash_{\Box} s''' \bowtie s_2$ and $s_1'' = s_1'$ and we are done, since $s_2 \bullet s_1 = (s'' \bullet s)^{\alpha}\backslash^{\Box}_{a}\backslash^{a}_{\Box}\backslash_{\beta}(s''' \bullet s') \rtimes\ltimes (s'' \bullet s)^{\alpha}\backslash^{a}_{a}\backslash_{\beta}(s' \bullet s''') = s_2' \bullet s_1''$. In the case (b) we take $s_2' = s_2$ and $s_1'' = s^{\alpha}\backslash^{\Box}_{a}\backslash^{a}_{\Box}\backslash_{\beta} s' \bowtie s_1'$ and we are done, since $s_2 \bullet s_1 = (s'' \bullet s)^{\alpha}\backslash^{a}_{a}\backslash^{a}_{a}\backslash_{\beta}(s''' \bullet s') \rtimes\ltimes (s'' \bullet s)^{\alpha}\backslash^{a}_{a}\backslash^{a}_{a}\backslash_{\beta}(s' \bullet s''') = s_2' \bullet s_1''$.

□

We recall Lemma 37 (the original appears on p. 29).

Lemma 37. *Let s and s' be two link chains such that $s \rtimes\ltimes s'$, then for any renaming function $s[\phi] \rtimes\ltimes s'[\phi]$.*

Proof. The proof is similar to the ones of Lemma 19 (point v) since, by definition $\phi(\tau) = \tau$ and $\phi(\Box) = \Box$, then the equivalence relation $\rtimes\ltimes$ is not affected by ϕ. □

We recall Theorem 42 (the original appears on p. 30).

Theorem 42. *Network bisimilarity is a congruence.*

Proof. We complete here the proof outlined at p. 30, by giving the details of the case for recursion.

Recursion Let E and F be two processes that invoke the process identifier X. Assume that for any process P we have $E\{P/X\} \bowtie F\{P/X\}$. We want to prove that, given the process definitions $A \triangleq E\{A/X\}$, $B \triangleq F\{B/X\}$, then $A \bowtie B$. The proof proceeds by showing that:

1. If $A \triangleq Q$ is a process definition, then $A \bowtie Q$.
2. Given the process definitions $A \triangleq E\{A/X\}$ and $B \triangleq F\{B/X\}$, for any process G that invokes X we have $G\{A/X\} \bowtie G\{B/X\}$.

Then, we have $A \bowtie E\{A/X\} \bowtie E\{B/X\} \bowtie F\{B/X\} \bowtie B$. The proof of (1) is immediate by rule *(Ide)*, as A and Q have exactly the same transitions, while the proof of (2) proceeds in the standard way exploiting induction on derivations as detailed below.

Let $\mathbf{R}_{ctx} \triangleq \{(G\{A/X\}, G\{B/X\}) \mid G \text{ is a process that possibly invokes } X\}$. Let $\mathbf{R}_{upto} \triangleq \bowtie \circ \mathbf{R}_{ctx} \circ \bowtie$. Note that $\mathbf{R}_{ctx}$ includes the identity relation when taking G with no occurrence of X. Moreover, $\mathbf{R}_{ctx} \subseteq \mathbf{R}_{upto}$, $\bowtie \subseteq \mathbf{R}_{upto}$ and $\mathbf{R}_{upto} \circ \bowtie = \mathbf{R}_{upto}$ because $\bowtie$ is an equivalence relation (and thus transitively closed). We prove that $\mathbf{R}_{upto}$ is a network bisimulation. To this aim, it is enough to consider a generic pair of processes $(G\{A/X\}, G\{B/X\})$ in $\mathbf{R}_{ctx}$ and prove that whenever $G\{A/X\} \xrightarrow{s} P'$ then there are some s' and Q' such that $G\{B/X\} \xrightarrow{s'} Q'$, $\mathsf{e}(s) = \mathsf{e}(s')$ and $(P', Q') \in \mathbf{R}_{upto}$. We proceed by induction on the derivation of the transition $G\{A/X\} \xrightarrow{s} P'$, by considering the possible shapes of G.

$G = X$**:** We have $G\{A/X\} = A$. Since $A \xrightarrow{s} P'$ and $A \triangleq E\{A/X\}$, it means that $E\{A/X\} \xrightarrow{s} P'$ with a shorter derivation than $A \xrightarrow{s} P'$. Hence, by inductive hypothesis, there are s'' and Q'' such that

$E\{B/X\} \xrightarrow{s''} Q''$, $\mathsf{e}(s) = \mathsf{e}(s'')$ and $(P', Q'') \in \mathbf{R}_{upto}$. Since $E\{P/X\} \bowtie\!\!\!\sim F\{P/X\}$ for any process P, we have in particular $E\{B/X\} \bowtie\!\!\!\sim F\{B/X\}$. So there are s' and Q' such that $F\{B/X\} \xrightarrow{s'} Q'$, $\mathsf{e}(s'') = \mathsf{e}(s')$ and $Q'' \bowtie\!\!\!\sim Q'$. Since $B \triangleq F\{B/X\}$, by applying rule *(Ide)* we have $B \xrightarrow{s'} Q'$. We conclude by noting that $G\{B/X\} = B$, $\mathsf{e}(s) = \mathsf{e}(s'') = \mathsf{e}(s')$ and that $(P', Q') \in \mathbf{R}_{upto} \circ \bowtie\!\!\!\sim = \mathbf{R}_{upto}$.

$G = \ell.G'$**:** We have $G\{A/X\} = \ell.(G'\{A/X\})$ and thus $P' = G'\{A/X\}$. Moreover $G\{B/X\} = \ell.(G'\{B/X\}) \xrightarrow{s} G'\{B/X\}$ and by definition of $\mathbf{R}_{upto}$ we have $(G'\{A/X\}, G'\{B/X\}) \in \mathbf{R}_{ctx} \subseteq \mathbf{R}_{upto}$.

$G = G_1 + G_2$**:** We have $G\{A/X\} = G_1\{A/X\} + G_2\{A/X\}$. Since we have $G\{A/X\} \xrightarrow{s} P'$ there are two possibilities: either $G_1\{A/X\} \xrightarrow{s} P'$ or $G_2\{A/X\} \xrightarrow{s} P'$ (with shorter derivations). Without loss of generality, let us consider just the first case. By inductive hypothesis, there are s' and Q' such that $G_1\{B/X\} \xrightarrow{s'} Q'$, $\mathsf{e}(s) = \mathsf{e}(s')$ and $(P', Q') \in \mathbf{R}_{upto}$. Then, by rule *(Lsum)*, $G\{B/X\} = G_1\{B/X\} + G_2\{B/X\} \xrightarrow{s'} Q'$.

$G = (\nu\, a)G'$**:** We have $G\{A/X\} = (\nu\, a)(G'\{A/X\})$. Thus $P' = (\nu\, a)P''$ and $s = (\nu\, a)s''$ for some P'' and s'' such that $G'\{A/X\} \xrightarrow{s''} P''$ (with a shorter derivation). By inductive hypothesis, there are s_1'', Q'' such that $G'\{B/X\} \xrightarrow{s_1''} Q''$, $\mathsf{e}(s'') = \mathsf{e}(s_1'')$ and $(P'', Q'') \in \mathbf{R}_{upto}$. By Lemma 35, there exists $s_2'' \bowtie s_1''$ such that $(\nu\, a)s_2'' \neq \bot$ and $(\nu\, a)s_2'' \rhd\!\lhd (\nu\, a)s''$. By the Accordion Lemma 25, $G'\{B/X\} \xrightarrow{s_2''} Q''$. Then, we take $s' = (\nu\, a)s_2''$ and $Q' = (\nu\, a)Q''$ and by rule *(Res)* $G\{B/X\} = (\nu\, a)(G'\{B/X\}) \xrightarrow{s'} Q'$. Clearly $\mathsf{e}(s) = \mathsf{e}(s')$. To see that $(P', Q') \in \mathbf{R}_{upto}$ we note that by $(P'', Q'') \in \mathbf{R}_{upto}$ there is some H such that $P'' \bowtie\!\!\!\sim H\{A/X\}$ and $H\{B/X\} \bowtie\!\!\!\sim Q''$. Then, as $\bowtie\!\!\!\sim$ is a congruence w.r.t. restriction, $P' = (\nu\, a)P'' \bowtie\!\!\!\sim (\nu\, a)H\{A/X\}$ and $(\nu\, a)H\{B/X\} \bowtie\!\!\!\sim (\nu\, a)Q'' = Q'$ and we are done.

$G = G'[\phi]$**:** This case in analogous to the previous one and thus omitted.

$G = G_1|G_2$**:** We have $G\{A/X\} = G_1\{A/X\}|G_2\{A/X\}$. Since $G\{A/X\} \xrightarrow{s}$

P' we have three cases: (i) $G_1\{A/X\} \xrightarrow{s} P'_1$ and $P' = P'_1|G_2\{A/X\}$, or (ii) $G_2\{A/X\} \xrightarrow{s} P'_2$ and $P' = G_1\{A/X\}|P'_2$, or (iii) $G_1\{A/X\} \xrightarrow{s_1} P'_1$, $G_2[\{A/X\} \xrightarrow{s_2} P'_2$ and $P' = P'_1|P'_2$ with $s = s_1 \bullet s_2$. Without loss of generality, let us focus on the third case, which is the more involved. By inductive hypothesis, there are s''_i, Q'_i with $G_i\{B/X\} \xrightarrow{s''_i} Q'_i$, $\mathtt{e}(s_i) = \mathtt{e}(s''_i)$ and $(P'_i, Q'_i) \in \mathbf{R}_{upto}$ for $i = 1, 2$. By Lemma 36, we know that s''_1 and s''_2 can be stretched respectively to $s'_1 \blacktriangleright\!\blacktriangleleft s''_1$ and $s'_2 \blacktriangleright\!\blacktriangleleft s''_2$ so that $s'_1 \bullet s'_2$ is defined and $\mathtt{e}(s_1 \bullet s_2) = \mathtt{e}(s'_1 \bullet s'_2)$. By the Accordion Lemma 25, $G_i\{B/X\} \xrightarrow{s'_i} Q'_i$ for $i = 1, 2$ and by rule *(Par)*, we get $G\{B/X\} = G_1\{B/X\}|G_2\{B/X\} \xrightarrow{s'} Q'$ with $s' = s'_1 \bullet s'_2$ and $Q' = Q'_1|Q'_2$. To see that $(P', Q') \in \mathbf{R}_{upto}$ we note that, for $i = 1, 2$, by $(P'_i, Q'_i) \in \mathbf{R}_{upto}$ there is some H_i such that $P'_i \bowtie H_i\{A/X\}$ and $H_i\{B/X\} \bowtie Q'_i$. Then, as $\bowtie$ is a congruence w.r.t. parallel composition, $P' = P'_1|P'_2 \bowtie H_1\{A/X\}|H_2\{A/X\}$ and $H_1\{B/X\}|H_2\{B/X\} \bowtie Q'_1|Q'_2 = Q'$ and we are done.

$G = C$**:** The simplest case is when G is a constant C associated with a definition $C \triangleq R$. In fact, we have $G\{A/X\} = C = G\{B/X\}$ and we conclude by taking $s' = s$ and $Q' = P'$.

□

We recall Lemma 44 (the original appears on p. 33).

Lemma 44. *For any a, b, s, s'*

(i) If $s \blacktriangleright\!\blacktriangleleft s'$ then $s\{b/a\} \blacktriangleright\!\blacktriangleleft s'\{b/a\}$.

(ii) If $s \bowtie s'$ then $s\{b/a\} \bowtie s'\{b/a\}$.

Proof. The proof proceeds by cases on the axioms of $\blacktriangleright\!\blacktriangleleft$ and $\bowtie$, see Definitions 8 and **??**. We prove only one case, the remaining ones are similar.

case $[s^{\alpha}\backslash^{\tau}_{\tau}\backslash_{\beta} s' \bowtie s^{\alpha}\backslash_{\beta} s']$ We have $s_1 = s^{\alpha}\backslash^{\tau}_{\tau}\backslash_{\beta} s'$ and $s_2 = s^{\alpha}\backslash_{\beta} s'$. Let a, b

two channel names then, by definition of substitution, we have

$$\begin{aligned} s_1\{b/a\} &= (s^{\alpha}\backslash^{\tau}_{\tau}\backslash_{\beta} s')\{b/a\} \\ &= (s[\{b/a\})\ ({}^{\alpha}\backslash_{\tau}\{b/a\})\ ({}^{\tau}\backslash_{\beta}\{b/a\})\ (s'\{b/a\}) \\ &= (s\{b/a\})\ {}^{\alpha\{b/a\}}\backslash^{\tau}_{\tau}\backslash_{\beta\{b/a\}}\ (s'\{b/a\}) \\ &\bowtie (s\{b/a\})\ {}^{\alpha\{b/a\}}\backslash_{\beta\{b/a\}}\ (s'\{b/a\}) \\ &= (s^{\alpha}\backslash_{\beta} s')\{b/a\} \\ &= s_2\{b/a\}. \end{aligned}$$

□

We recall Lemma 51 (the original appears on p. 38).

Lemma 51. *Let $R(\widetilde{a},\widetilde{b})$ be a composite infrastructure.*

1. *If $R(\widetilde{a},\widetilde{b}) \xrightarrow{s} R'$ then $R' = R(\widetilde{a},\widetilde{b})$ and there exist two nodes $a_i \in \widetilde{a}$ and $b_j \in \widetilde{b}$ of the graph $\mathcal{G}(R(\widetilde{a},\widetilde{b}))$ such that $s \bowtie {}^{a_i}\backslash_{b_j}$ and there is a path from a_i to b_j whose length is $||s||$ in the graph $\mathcal{G}(R(\widetilde{a},\widetilde{b}))$.*
2. *If there is a path of length n from a_i to b_j in the graph $\mathcal{G}(R(\widetilde{a},\widetilde{b}))$ then $R(\widetilde{a},\widetilde{b}) \xrightarrow{s} R(\widetilde{a},\widetilde{b})$ with $s \bowtie {}^{a_i}\backslash_{b_j}$ and $||s|| = n$.*

Proof. We prove the two implications separately.

1. The proof is by structural induction on the composite infrastructure $R(\widetilde{a},\widetilde{b})$. If it is a basic infrastructure $R(\widetilde{a},\widetilde{b}) = \ell_1.R(\widetilde{a},\widetilde{b}) + ... + \ell_k.R(\widetilde{a},\widetilde{b})$, then it must be the case that $R' = R(\widetilde{a},\widetilde{b})$ and $s \blacktriangleright\!\blacktriangleleft \ell_h = {}^{a_{i_h}}\backslash_{b_{j_h}}$ for some $h \in [1,k]$. Clearly $||s|| = 1$ and in fact there is a path of length 1 from a_{i_h} to b_{j_h} in the graph $\mathcal{G}(R(\widetilde{a},\widetilde{b}))$.
   If it is the composition

$$(\nu\,\widetilde{c})(Q(\widetilde{a},\widetilde{c}) \mid S(\widetilde{c},\widetilde{b}))$$

   of two infrastructures, then it must be the case that $s = (\nu\,\widetilde{c})(s_1 \bullet s_2)$ for some s_1 and s_2 such that there exists Q' and S' with $Q(\widetilde{a},\widetilde{c}) \xrightarrow{s_1} Q'$, $S(\widetilde{c},\widetilde{b}) \xrightarrow{s_2} S'$ and $R' = (\nu\,c)(Q' \mid S')$. Then, by inductive hypotheses, we know that

- $Q' = Q(\widetilde{a}, \widetilde{c})$ and there exist two nodes $a_i \in \widetilde{a}$ and $c_h \in \widetilde{c}$ of the graph $\mathcal{G}(Q(\widetilde{a}, \widetilde{c}))$ (and thus also in $\mathcal{G}(R(\widetilde{a}, \widetilde{b}))$) such that $s_1 \rhd\!\lhd {}^{a_i}\backslash_{c_{h_1}}$ and there is a path from a_i to c_{h_1} whose size is $||s_1||$.
- $S' = S(\widetilde{c}, \widetilde{b})$ and there exist two nodes $c_{h_2} \in \widetilde{c}$ and $b_j \in \widetilde{b}$ of the graph $\mathcal{G}(S(\widetilde{c}, \widetilde{b}))$ (and thus also in $\mathcal{G}(R(\widetilde{a}, \widetilde{b}))$) such that $s_2 \rhd\!\lhd {}^{c_{h_2}}\backslash_{b_j}$ and there is a path from c_{h_2} to b_j whose length is $||s_2||$.

Since channels $\widetilde{c}$ are restricted and $(\nu\, c)(s_1 \bullet s_2)$ is well defined, it must be the case that $h_1 = h_2$ and $s_1 \bullet s_2 \rhd\!\lhd {}^{a_i}\backslash^{c_{h_1}}_{c_{h_1}}\backslash_{b_j}$. Therefore $R' = (\nu\, \widetilde{c})(Q' \mid S') = (\nu\, \widetilde{c})(Q(\widetilde{a}, \widetilde{c})|S(\widetilde{c}, \widetilde{b})) = R(\widetilde{a}, \widetilde{b})$, $s = (\nu\, \widetilde{c})(s_1 \bullet s_2) \rhd\!\lhd {}^{a_i}\backslash_{b_j}$, $||s|| = ||s_1|| + ||s_2||$ and the two paths from a_i to c_{h_1} and from c_{h_1} to b_j can be composed to form a path from a_i to b_j whose length is exactly $||s||$.

2. The proof is by structural induction on the composite infrastructure $R(\widetilde{a}, \widetilde{b})$.
   If it is a basic infrastructure $R(\widetilde{a}, \widetilde{b}) = \ell_1.R(\widetilde{a}, \widetilde{b}) + ... + \ell_k.R(\widetilde{a}, \widetilde{b})$ then the path from a_i to b_j in the graph must have length one and be in correspondence to one of the links offered by $R(\widetilde{a}, \widetilde{b})$.
   If it is the composition

$$(\nu\, \widetilde{c})(Q(\widetilde{a}, \widetilde{c}) \mid S(\widetilde{c}, \widetilde{b}))$$

   of two infrastructures, then it must be the case that the path from a_i to b_j with length n can be split in two parts: from a_i to some c_h (contained in the graph $\mathcal{G}(Q(\widetilde{a}, \widetilde{c}))$) and from c_h to b_j (contained in the graph $\mathcal{G}(S(\widetilde{c}, \widetilde{b}))$), respectively with lengths n_1 and n_2 such that $n = n_1 + n_2$. Then, by the inductive hypotheses, it must be the case that

   - $Q(\widetilde{a}, \widetilde{c}) \xrightarrow{s_1} Q(\widetilde{a}, \widetilde{c})$ with $s_1 \rhd\!\lhd {}^{a_i}\backslash_{c_h}$ and $||s_1|| = n_1$.
   - $S(\widetilde{c}, \widetilde{b}) \xrightarrow{s_2} S(\widetilde{c}, \widetilde{b})$ with $s_2 \rhd\!\lhd {}^{c_h}\backslash_{b_j}$ and $||s_2|| = n_2$.

   Then we can find two suitable chains $s_1' \blacktriangleright\!\blacktriangleleft s_1$ and $s_2' \blacktriangleright\!\blacktriangleleft s_2$ such that $s_1' \bullet s_2'$ is well defined and $s_1' \bullet s_2' \rhd\!\lhd {}^{a_i}\backslash^{c_h}_{c_h}\backslash_{b_j}$. Therefore we take $s = (\nu\, \widetilde{c})(s_1' \bullet s_2') \rhd\!\lhd {}^{a_i}\backslash_{b_j}$ with $||s|| = ||s_1'||+||s_2'|| = ||s_1||+||s_2|| = n_1+n_2 = n$ and by the rules of the operational semantics we have $R(\widetilde{a}, \widetilde{b}) \xrightarrow{s} R(\widetilde{a}, \widetilde{b})$.

□