Proc. of *ICGT'08*, LNCS 5214, pages 131–146. © Springer.

Network Applications of Graph Bisimulation

Pietro Cenciarelli[1], Daniele Gorla[1] and Emilio Tuosto[2]

[1] Dip. di Informatica, "Sapienza" Università di Roma (Italy)
[2] Dept. Computer Science, Univerity of Leicester (UK)

Abstract *Synchronising Graphs* is a system of parallel graph transformation designed for modeling process interaction in a network environment. We propose a theory of *context-free* synchronising graphs and a novel notion of bisimulation equivalence which is shown to be a congruence with respect to graph composition and node restriction. We use this notion of equivalence to study some sample network applications, and show that our bisimulation equivalence captures notions like functional equivalence of logical switches, equivalence of channel implementations and level of fault tolerance of a network.

1 Introduction

Synchronising Graphs (SG) is a system of parallel graph transformation designed for modeling process interaction in a network environment. The system is inspired by [8], and it stems from the *Synchronized Hyperedge Replacement* (SHR) of [10], with which it has been compared in [4]. In the SG model, as in SHR, hyperedges represent agents, or software components, while nodes are thought of as communication channels, synchronisation points or, more generally, network communication infrastructure. The idea that hypergraphs may interact by synchronising action and co-action pairs at specific synchronisation points (the nodes) is quite intuitive, while the flexibility of the model in representing diverse network topologies and communication protocols makes SG fit as a common semantic framework for interpreting different calculi. We followed this idea in [3], where *Mobile Ambients* [2] and the *distributed CCS* of [22] (without restriction) were both modeled in SG by using a common recursive architecture.

Here, we explore an orthogonal issue, namely the *behavioural equivalence* of SG. Indeed, such equivalences are often sought in the theory of concurrency for proving the conformance of an implementation with respect to a specification or for achieving a sort of compositionality in the semantics. If we identify the meaning of a process in its abstract behaviour (which is traditionally considered its bisimulation equivalence class), compositionality requires that, when equivalent processes (e.g. a specification and an implementation) are plugged into the same context, they behave in the same way. This amounts to proving that bisimulation equivalence is a congruence. However, although such results are abundant in the literature for process calculi, not so for graph rewriting, where system behaviour is typically context dependent.

To our knowledge, the most strictly related notions of behavioural equivalence proposed for related systems of graph transformation are [13] and [14]. The first paper proposes a behavioural equivalence for a model called *synchronised graph rewriting*; as pointed out by the authors, this equivalence is rather coarse, in that it is not able to distinguish graphs with different degrees of parallelism. In this paper, we meet their challenge for a finer notion and propose one capable of detecting parallelism (it is indeed possible to implement in SG Plotkin's parallel or). The behavioural equivalence of [14] refers to a system of graph rewriting, the SHR of [10], which differs from ours in the mathematical presentation of graphs, in their LTS and, more importantly, in the proof theory. Syntax is pervasive in SHR, which is more deeply rooted in the field of process calculi, of which it shares notions such as structural congruence and name binding. Nodes are treated as names are in process calculi. Unlike in SG, no semantic difference can be made between two nodes beside them being distinct. Not always so in graph rewriting, where transformations may depend on attachment to specific nodes. As shown in Section 3 (example 5), such dependency may cause non-compositional behaviour. Hence, while compositionality is to be expected in hyperedge replacement [14], not so for SG, which allows, as many graph rewrinting systems [7], context-dependent specifications. Thus, we characterise the theories of synchronising graphs, called *context-free* where compositionality holds. A natural notion of bisimulation equivalence is introduced to capture their abstract behaviour, and proven a congruence in any context-free theory. A similar result is presented in [14] for hyperedge replacement by exploiting the sytactic presentation of graphs and referring to results obtained in [24] in the context of structural operational semantics. Here we provide a *direct* proof, which relies on no syntax and sheds light on the meta-theoretical properties of our system (Lemmas 1 and 2). While imposing on axiom formats built-in features of SHR, our result is no special case of that in [14], as discussed in the conclusions.

Then, we use our framework for modeling four network applications where the proposed notion of bisimulation equivalence is shown to capture interesting properties. In the first application we consider network implementations of *logical switches*. Here bisimulation equivalence corresponds to functional equivalence, in that equivalent networks have an identical input-output behaviour. Then, we consider network implementations of *communication channels*, where information items can travel in parallel. Bisimulation equivalence is shown to capture the notion of (static) channel capacity. In the third scenario, we refine the previous model by introducing *node charges*, that are consumed upon passage of information. This feature cruises in several wireless applications and becomes a crucial issue in "extreme" applications like the *Smart Dust* [25]. Here bisimulation equivalence implies identity of dynamic capacity but provides a finer notion of observational equivalence which can be employed for net optimisation. Finally, we study the impact of *failure* in a communication net and show that bisimulation equivalence characterises exactly the degree of *fault tolerance*, or *robustness*, of a net.

The paper is structured as follows. In Section 2, we present the general model of SG. In Section 3 we investigate the possible sources of context dependency in SG and focus on *context-free* SG; a novel notion of bisimulation equivalence is then introduced and shown to be a congruence. Section 4 presents the applications. Section 5 concludes the paper by discussing related work and by hinting at current and future research. For space reasons, all proofs are moved to the appendix.

2 Synchronising Graphs

Let $\mathcal{N}$ be a set of *nodes*, which we consider fixed throughout. A *graph* (E, G, R) consists of a set E of *hyperedges*, an attachment function $G : E \to \mathcal{N}^*$ and a set $R \subseteq |G|$ of nodes, called *restricted*, where $|G| = \{x \in \mathcal{N} \mid \exists\, e \in E \text{ s.t. } x \in G\,e\}$ is the set of nodes of the graph. When clear from the context or when not important, we shall write a graph by simply specifying its attachment function. When $G\,e = x_1 x_2 \ldots x_n$ (we shall often abbreviate $x_1 x_2 \ldots x_n$ as $\boldsymbol{x}$), we call n the *arity* of e and say that the i-th *tentacle* of e is attached to x_i. We denote by $res\,(G)$ the set of restricted nodes of G, and by $fn\,(G)$ the set $|G| - res\,(G)$ of *free* nodes. We write $e(\boldsymbol{x})$ for an hyperedge such that $G\,e = \boldsymbol{x}$.

Let $Act = \{a, b, \ldots\} \cup \{\overline{a}, \overline{b}, \ldots\}$ be a set of *actions*; we call $\overline{a}$ the *co-action* of a, and intend a by $\overline{\overline{a}}$. A *pre-transition* is a triple (G, Λ, H), written $G \xrightarrow{\Lambda} H$ (or just Λ for short), where $\Lambda \subseteq \mathcal{N} \times Act \times \mathcal{N}^*$ is a relation, while G and H are graphs, called respectively the *source* and the *destination* of Λ. Intuitively, $(x, a, \boldsymbol{y}) \in \Lambda$ expresses the occurrence of action a at node x, which can be thought as a communication channel, while the elements of $\boldsymbol{y}$, called *objects*, are thought of as arguments. When $\boldsymbol{y}$ is the empty sequence ϵ, (x, a, ϵ) is written (x, a).

In SG the occurrence of both $(x, a, \boldsymbol{y})$ and $(x, \overline{a}, \boldsymbol{z})$ in Λ is called a *synchronisation*, and it corresponds to the *silent* action τ of most process calculi. Synchronising hyperedges may exchange information. This is implemented in SG by unifying the lists $\boldsymbol{y}$ and $\boldsymbol{z}$ of objects, which are required to be of the same length. Only two agents at a time may synchronise at one node. Moreover, if an action occurs at a restricted node, then it *must* synchronise with a corresponding co-action, as we consider *observable* the unsynchronised actions. A restricted node may be "opened" by unifying it with an argument of an observable action, or with a node which is not restricted.

Notation If $\varphi \subseteq A \times B$ is a relation and $a \in A$, we write $\varphi\, a$ the set $\{b \in B : (a, b) \in \varphi\}$. The *domain* of φ is the set $dom\,(\varphi) = \{a \in A : \exists\, b \in B\,.\,(a, b) \in \varphi\}$. A function $f : A \to B$ is said to *agree* with φ when $f\,x \in \varphi\, x$, for all $x \in A$. If φ is an equivalence relation, $[x]_\varphi$ is the equivalence class of an element x, which we write $[x]$ when φ is understood. A *unifier* of φ is a function f which agrees with φ as above and such that $f[x]$ is a singleton, for all x.

If $f : \mathcal{N} \to \mathcal{N}$ is a function on nodes and (E, G, R) is a graph, we write fG the graph (E, fG, fR) obtained by substituting all nodes x in G with $f\,x$. More precisely, for all $e \in E$, if $G\,e = x_1 \ldots x_n$ then $(fG)\,e = f\,x_1 \ldots f\,x_n$.

Given a pre-transition $G \xrightarrow{\Lambda} H$, we denote by $|\Lambda|$ the set $|G| \cup |H|$ and by $res\,(\Lambda)$ the set $res\,(G) \cup res\,(H)$. By $obj\,(\Lambda)$ we denote the set $\{y \in \mathcal{N} : \exists\,(x, a, \boldsymbol{y}) \in \Lambda \text{ such that } y \in \boldsymbol{y}\}$. We omit parentheses and braces when listing the elements of Λ above a transition arrow.

An *action set* is a relation $\Lambda \subseteq \mathcal{N} \times Act \times \mathcal{N}^*$ such that, for all nodes x, $\Lambda\, x$ has *at most* two elements and, when so, it is of the form $\{(a, \boldsymbol{y}), (\overline{a}, \boldsymbol{z})\}$, where $\boldsymbol{y}$ and $\boldsymbol{z}$ are vectors of identical length. Given an action set Λ, we denote by $\stackrel{\Lambda}{=}$ the smallest equivalence relation on nodes such that, if $(x, a, y_1 y_2 \ldots y_n)$ and $(x, \overline{a}, z_1 z_2 \ldots z_n)$ are in Λ, then $y_i \stackrel{\Lambda}{=} z_i$, for $i = 1 \ldots n$. By a slight abuse, we say that a function agrees with (or unifies) an action set Λ to mean that it agrees with (unifies) the relation $\overset{\Lambda}{=}$. Arguments of unsynchronised actions are called *dangling*. More precisely, we call dangling in Λ the elements of the set $dng\,(\Lambda) = \{z \in obj\,(\Lambda) \;:\; \Lambda\, x = \{(a, \boldsymbol{y})\}$ and $z \overset{\Lambda}{=} y,$ for some x and $y \in \boldsymbol{y}\}$.

Definition 1. *A* transition *is a pre-transition* $G \overset{\Lambda}{\to} H$ *such that:*

1. *Λ is an action set such that* $\mathsf{dom}\,(\Lambda) \cup obj\,(\Lambda) \subseteq |G|$*;*
2. *if a node x is restricted in G then $\Lambda\, x$ is not a singleton;*
3. *if $x \in |H|$, then $x \in fn\,(H)$ if and only if $x \in fn\,(G) \cup dng\,(\Lambda)$.*
4. *$H = \rho H$ for some unifier ρ of Λ such that $\rho\, x \in fn\,(G)$ for all $x \in fn\,(G)$.*

Condition 1 expresses the *locality* of action: graphs can only act upon their own nodes. By this condition, for example, the pre-transition $e(x) \xrightarrow{x,a,y} d(y)$, legal in SHR, is not a transition, because $y \notin |e(x)|$. A consequence of 1 and 3 is that all free nodes in the destination of a transition must occur in the source. Hence, while $e(x) \overset{\emptyset}{\to} \nu y\, d(y)$ is a legal transition, $e(x) \overset{\emptyset}{\to} d(y)$ is not. This rules *ownership* of nodes: the access to a new channel is only acquired via synchronisation. Condition 4 enforces fusions. It also grants a privilege to the free nodes when they are fused with the bound, which allows $\nu y\, e(x\, y) \xrightarrow{\substack{x,\overline{a},x \\ x,a,y}} d(x)$ and forbids $\nu y\, e(x\, y) \xrightarrow{\substack{x,\overline{a},x \\ x,a,y}} d(y)$. This restriction is not essential for the theory of synchronising graphs while it simplifies the meta-theory without loss of generality.

In SG, synchronisation is subject to a non-interference condition: two transitions can be synchronised provided they are disjoint and they share no restricted nodes. Formally, $G \overset{\Lambda}{\to} H$ and $F \overset{\Theta}{\to} K$ are said to be *non-interfering*, written $\Lambda \# \Theta$, whenever $\Lambda \cap \Theta = \emptyset$ and $res\,(\Lambda) \cap |\Theta| = res\,(\Theta) \cap |\Lambda| = \emptyset$. It is an easy check that the only nodes two non-interfering transitions may have in common are the free nodes in their sources.

The rules of the system of synchronising graphs are given below. The *composite* of two graphs (E, G, R) and (D, F, S), written $G|F$, is defined when E and D are disjoint and moreover $res\,(G) \cap |F| = res\,(F) \cap |G| = \emptyset$; when so, $G|F$ is the graph $(E \cup D, G + F, R \cup S)$, where $G + F$ is the attachment function mapping $e \in E$ to $G\, e$ and $d \in D$ to $F d$. We let $\nu x\, G$ denote the graph $(E, G, R \cup \{x\})$ when $x \in |G|$, while $\nu x\, G = G$ otherwise.

$$[\,\text{sync}\,]\ \frac{G \overset{\Lambda}{\to} H \qquad F \overset{\Theta}{\to} K}{G|F \xrightarrow{\Lambda\cup\Theta} \rho(H|K)}\ \Lambda \# \Theta \text{ and } \rho \text{ unifies } \Lambda \cup \Theta$$

$$[\,\text{open}\,]\ \frac{G \overset{\Lambda}{\to} H}{\nu x\, G \overset{\Lambda}{\to} H}\ x \in dng\,(\Lambda) \qquad\qquad [\,\text{res}\,]\ \frac{G \overset{\Lambda}{\to} H}{\nu x\, G \overset{\Lambda}{\to} \nu x\, H}\ x \notin dng\,(\Lambda)$$

A *theory* of synchronising graphs is a set of transitions which is closed under the inference rules. The smallest theory including a given set $\mathcal{A}$ of transitions is said to be *generated* by the *axioms* in $\mathcal{A}$.

Note that inference rules assume, as implicit side condition, that the conclusion be a transition. Hence, for example, the rule [sync] does not apply to $\nu y\, e(x\, y) \xrightarrow{x,a,y} f(y)$

and $\nu z\, d(x\, z) \xrightarrow{x,\bar{a},z} g(z)$ because the conclusion would violate condition 3 of definition 1. Also note that, differently from the π-calculus [19], we do not have a "close" rule to close the scope of a restricted name after having opened it via an "open" rule. This is related to the fact that every inference in SG can be rewritten in a sort of 'normal form' where all the applications of [res] and [open] needed to infer the judgement come after all the applications of [sync] (see Lemmata 1 and 2 later on). This is similar to the presentations of the LTS for the π-calculus that include structural equivalence: in those cases, a "close" rule is omitted because redundant. Thus, for example, we can build an inference for the graph $\nu xG \mid \nu yH$ where the two parallel components synchronise, assuming that $G \xrightarrow{z,a,x} G'$ and that $H \xrightarrow{z,\bar{a},y} H'$: indeed, $\nu xG \mid \nu yH$ is just another (but exactly identical) way of writing the graph $\nu x(\nu y(G \mid H))$, that reduces to, e.g., $\nu x(\nu y(G' \mid H'\{x/y\}))$ after a transition $\{(z, a, x), (z, \bar{a}, y)\}$.

Example 1 (A non-deterministic commuter). Consider a system consisting of several input and output *sockets*. The system, which we shall call *non-deterministic commuter*, acts by non-deterministically connecting client processes (possibly attached to an input socket) with one of the output sockets (where server processes may be attached). Connections are established one at a time. Figure 1 depicts a commuter **C** with three input and two output sockets. A client process **r** is being connected with a server **q**.

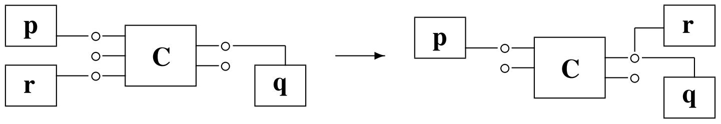

Figure 1. A non-deterministic commuter

Non-deterministic commuters can be engineered in SG by assembling simple components (edges) of the form $in\,(x\,u)$ and $out\,(u\,y)$, representing input and output sockets respectively. Clients are meant to be attached to the x node of a socket, while servers are attached to y. The node u represents an internal communication channel of the system. As elsewhere in the paper, we may use the same name to denote distinct edges representing components of the same kind. For example, $in\,(x\,u) \mid in\,(z\,u)$ will denote a graph with *two* edges, each representing an input socket. Then, ignoring the two unused sockets of **C** (viz., the second input socket and the second output socket), the initial state of the commuter is represented by the graph $\nu u\,(in\,(x\,u) \mid in\,(z\,u) \mid out\,(u\,y))$. The system's behaviour is specified by the following two transitions, where a and $\overline{a}$ represent the input and output actions respectively:

$$in\,(x\,u) \xrightarrow{u,a,x} \emptyset \qquad\qquad out\,(u\,y) \xrightarrow{u,\bar{a},y} out\,(u\,y)$$

To be precise, these are to be considered as axiom *schemes*, and we assume one axiom of the first kind for each input socket and one of the second for each output. In the present example, we further assume that any hyperedge can perform a passive (empty) transition to itself. Then, ignoring **p** and its socket, the transition of figure 1 is obtained

as $\quad r(z) \mid q(y) \mid \nu u\,(in\,(z\,u) \mid out\,(u\,y)) \xrightarrow{\substack{u,\bar{a},y\\ u,a,z}} r(y) \mid q(y) \mid \nu u\, out\,(u\,y).$ $\square$

To conclude the presentation of SG, we give two meta-theoretical lemmata showing that any transition in a given theory can be inferred by a canonical derivation where all applications of [sync] precede [res] and [open].

Notation Let $G \xrightarrow{\Lambda} H$ and $F \xrightarrow{\Theta} K$ be transitions; we denote by $\Lambda * \Theta$ the set of transitions of the form $G|F \xrightarrow{\Lambda \cup \Theta} \rho(H|K)$ obtained by synchronising Λ and Θ with [sync]. Clearly, $\Lambda * \Theta$ is empty when Λ and Θ interfere. The expression $(\Lambda * \Theta) * \Phi$ stands for $\bigcup_{\Xi \in \Lambda * \Theta}(\Xi * \Phi)$. Similarly, we let $\nu x\, \Lambda$ be the transition which results from restricting Λ on x by an application of [res] or [open]. The expression $\nu x\, (\Lambda * \Theta)$ denotes the set of transitions of the form $\nu x\, \Xi$ with $\Xi \in \Lambda * \Theta$.

Lemma 1. *Let Λ and Θ be transitions and let x occur unrestricted in the source of Λ. Then, $(\nu x\, \Lambda) * \Theta \subseteq \nu x\, (\Lambda * \Theta)$.*

The opposite inclusion does not hold: $\nu y\, e(x\, y) \,|\, \nu z\, d(x\, z) \xrightarrow{\substack{x,\bar{a},z \\ x,a,y}} \nu y\, (h(y) \,|\, k(y))$ is included in $\nu y\, (\Lambda * \Theta)$ where Λ is $e(x\, y) \xrightarrow{x,a,y} h(y)$ and Θ is $\nu z\, d(x\, z) \xrightarrow{x,\bar{a},z} k(z)$, while $(\nu y\, \Lambda) * \Theta$ is empty because the result of applying [sync] to $\nu y\, \Lambda$ and Θ violates condition 3 of definition 1.

Lemma 2. *Synchronisation is associative: $(\Lambda * \Theta) * \Xi = \Lambda * (\Theta * \Xi)$.*

3 Context-free Theories and Behavioural Equivalence

One of the aims of the present paper is to characterise the theories of synchronising graphs in which the behaviour of a graph is not affected by the context. The following examples will clarify this concept.

Example 2. In the theory generated by a unique axiom $e|d \xrightarrow{\emptyset} \emptyset$, the two processes e and d, considered in isolation, have the same behaviour: none of them can move. However, if set in the context $[\,\_\,]|d$, the two processes exhibit quite different behaviour, as $e|d$ can move while $d|d$ cannot.

Example 3. In the theory generated by a unique axiom $\nu x\, e(x) \xrightarrow{\emptyset} \emptyset$, the process $e(x)$ cannot move, thus exhibiting the same catatonic behaviour as the empty process $\emptyset$. However, when set in a context $\nu x\, [\,\_\,]$ where x is restricted, $\nu x\, e(x)$ can move while $\nu x\, \emptyset = \emptyset$ cannot.

Example 4. In the theory generated by the four axioms $h(x\, y) \xrightarrow{\substack{x,\bar{a},x \\ x,a,y}} \emptyset$, $d \xrightarrow{\emptyset} d$, $e(x\, y) \xrightarrow{\emptyset} e(x\, y)$ and $e(x\, x) \xrightarrow{x,a} \emptyset$, $e(x\, y)$ behaves just like the process d, cycling forever over itself. However, when put in parallel with $h(x\, y)$, $e(x\, y)$ yields a trace which $h(x\, y) \,|\, d$ does not have: $h(x\, y)|e(x\, y) \xrightarrow{\substack{x,\bar{a},x \\ x,a,y}} e(x\, x) \xrightarrow{x,a} \emptyset$.

Example 5. In the theory generated by the three axioms $h(x\, y) \xrightarrow{\substack{x,\bar{a},x \\ x,a,y}} \emptyset$, $d \xrightarrow{\emptyset} d$ and $e(x) \xrightarrow{\emptyset} e(x)$, the processes $e(x)$ and d have the same behaviour. However, when put in parallel with $h(x\, y)$, $e(x)$ yields a transition to a catatonic state, namely $e(y)$, which $h(x\, y) \,|\, d$ cannot reach.

These are in fact the *only* possible sources of context dependency in a theory of synchronising graphs. This is shown in the present section by providing a notion of bisimulation equivalence on graphs and then proving that, in any theory generated by axioms including no transitions of the kind described in the examples, the proposed equivalence is a congruence with respect to restriction and parallel composition.

In this section we abandon the brute force notion of node substitution in a graph G adopted in the previous section and denote by hG the graph obtained by applying a substitution h to the *free* nodes of G, while restricted nodes are suitably renamed so as to avoid capture. This simplifies the treatment while remaining consistent with the theory developed so far. In particular, note that the new interpretation of $\rho(H|K)$ in [sync] does not alter the set of derivable transitions.

An *instance* of a transition $G \xrightarrow{\Lambda} H$ is a transition of the form $hG \xrightarrow{h\Lambda} \rho\, hH$ where h is a node substitution $\mathcal{N} \to \mathcal{N}$ and ρ is a unifier of $h\Lambda$. A *production* is a transition whose source consists of a single hyperedge $e(\boldsymbol{x})$, where all components of $\boldsymbol{x}$ are distinct and none of them is restricted. A theory of synchronising graphs is called *context-free* when it is generated by all the instances of a given set of productions. Note that the constraints that productions are asked to satisfy prevent the first three examples of context dependency to occur, while the use of *all* their instances for generating the theory accounts for the fourth example.

We now move to the definition of our behavioural equivalence; to this aim, we call *parameters* the elements of the set $\mathcal{P} = \mathcal{N} \times Act \times \mathbb{N}$. Intuitively, a parameter (x, a, i) is an abstraction over the i-th argument y_i of an action $(x, a, \boldsymbol{y})$. We call *observations* the elements of the set $\mathcal{O} = \mathcal{N} \cup \mathcal{P}$. Given an action set Λ, the relation $\overset{\Lambda}{=}$ extends to a relation $\overset{\Lambda}{=}_o$ on observations that is the smallest equivalence relation containing $\overset{\Lambda}{=}$ such that $(x, a, i) \overset{\Lambda}{=}_o (x, \overline{a}, i)$ and moreover $(y, b, j) \overset{\Lambda}{=}_o z$ if $(y, b, z_1 \ldots z_j \ldots z_n) \in \Lambda$ and $z = z_j$.

Not all pairs of $\overset{\Lambda}{=}_o$ are observable. The set $obs(\Lambda)$ of *observables* of a transition $G \xrightarrow{\Lambda} H$ consists of its observable nodes, the set of which we denote by $|\Lambda|_o$, together with the parameters of unsynchronised actions: $obs(\Lambda) = |\Lambda|_o \cup \{(x, a, i) \in \mathcal{P} \,:\, \Lambda\, x = \{(a, \boldsymbol{y})\} \text{ and } 0 \le i \le |\boldsymbol{y}|\}$, where $|\Lambda|_o = \{x \in fn(G) \,:\, x \in dom(\Lambda) \text{ or } x \text{ is dangling or } x \overset{\Lambda}{=} y \neq x \text{ for some } y \in fn(G)\}$. Note that, by the definition of $|\Lambda|_o$, while x is observable in $e(x\,y) \xrightarrow{\substack{x,\overline{a},y\\ x,a,y}} H$, y is not because, although it is free in $e(x\,y)$, "self-fusion" has no bearing on the interacting environment. The *observable part* of the relation $\overset{\Lambda}{=}_o$, written $\overset{\Lambda}{\simeq}$, is the equivalence relation obtained by restricting $\overset{\Lambda}{=}_o$ to $obs(\Lambda)$; thus, we let $p \overset{\Lambda}{\simeq} q$ if and only if $p \overset{\Lambda}{=}_o q$ and $p, q \in obs(\Lambda)$.

Definition 2. *Two transitions* $G \xrightarrow{\Lambda} H$ *and* $F \xrightarrow{\Theta} K$ *are called* equivalent *when* $\overset{\Lambda}{\simeq} = \overset{\Theta}{\simeq}$ *and* $\{x \in fn(G) \,:\, |\Lambda\, x| = 2\} = \{y \in fn(F) \,:\, |\Theta\, y| = 2\}$.

In our study of behavioural equivalence we follow a standard practice in process algebra where alpha-equivalent terms are considered as identical. In our context this amounts to defining behaviour on *classes* of alpha-equivalent graphs, that is graphs which are identical up to renaming of restricted nodes. We shall call such classes *abstract graphs*, and write them in bold, $\mathbf{G}$. Any theory of synchronising graphs yields a transition system of abstract graphs which includes $\mathbf{G} \xrightarrow{\Lambda} \mathbf{H}$ if and only if $G \xrightarrow{\Lambda} H$ is in the theory, for some $G \in \mathbf{G}$ and $H \in \mathbf{H}$.

Notice that the notion of free names can be extended to abstract graphs since, for every G and G' in $\mathbf{G}$, it holds that $fn(G) = fn(G')$; thus, the notion of equivalent transitions scales to abstract graphs as well. A transition $\mathbf{G} \xrightarrow{\Lambda} \mathbf{H}$ is said to be *fair* with an abstract graph $\mathbf{F}$ when none of the nodes in $obj(\Lambda) \setminus fn(\mathbf{G})$ is free in $\mathbf{F}$.

Definition 3. *A* simulation *is a binary relation $\mathcal{S}$ on abstract graphs such that* $\mathbf{G}\,\mathcal{S}\,\mathbf{F}$ *implies that, for all transitions* $\mathbf{G} \xrightarrow{\Lambda} \mathbf{H}$ *fair with* $\mathbf{F}$*, there exists a transition* $\mathbf{F} \xrightarrow{\Theta} \mathbf{K}$ *such that Λ and Θ are equivalent, and* $\mathbf{H}\,\mathcal{S}\,\mathbf{K}$*. An abstract graph* $\mathbf{G}$ *is* simulated *by a graph* $\mathbf{F}$*, written* $\mathbf{G} \prec \mathbf{F}$*, if there exists a simulation $\mathcal{S}$ such that* $\mathbf{G}\,\mathcal{S}\,\mathbf{F}$*. A* bisimulation *is a symmetric simulation. Two abstract graphs* $\mathbf{G}$ *and* $\mathbf{F}$ *are called* bisimulation equivalent*, written* $\mathbf{G} \sim \mathbf{F}$*, when they are related by a bisimulation.*

Notice that the fairness condition asked for $\mathbf{G} \xrightarrow{\Lambda} \mathbf{H}$ in the previous definition is standard in name-passing calculi, e.g. the π-calculus [19].

Composition and restriction extend to abstract graphs. In particular, $\nu x\,\mathbf{G}$ is $[\nu x\,G]_\alpha$, for some $G \in \mathbf{G}$ such that $x \notin res(G)$, while $\mathbf{G}|\mathbf{F}$ is $[G|F]_\alpha$, for some $G \in \mathbf{G}$ and $F \in \mathbf{F}$ such that $G|F$ is defined. Note that the above definitions do not depend on specific choices of G and F. A relation $\mathcal{R}$ on abstract graphs is called a *congruence* when $\mathbf{G}\,\mathcal{R}\,\mathbf{F}$ implies $\nu x\,\mathbf{G}\,\mathcal{R}\,\nu x\,\mathbf{F}$ and $\mathbf{G}|\mathbf{H}\,\mathcal{R}\,\mathbf{F}|\mathbf{H}$, for all x and $\mathbf{H}$.

Theorem 1. *Bisimulation equivalence is a congruence.*

Proof (Sketch). The result is proven by showing that the symmetric relation

$$\mathcal{R} = \{(\nu\boldsymbol{x}\,(\mathbf{G}|\mathbf{U}), \nu\boldsymbol{x}\,(\mathbf{F}|\mathbf{U})) \;:\; \mathbf{G} \prec \mathbf{F}\}$$

is a simulation. Then, closure under parallel composition is obtained by letting $\boldsymbol{x}$ be the empty vector; closure under name restriction is obtained by letting $\mathbf{U}$ be $[(\emptyset;\emptyset;\emptyset)]_\alpha$. Lemmas 1 and 2 are used. See appendix for detail. □

4 Network Applications

4.1 A non-deterministic commuter (Example 1 continued)

The internal communication channel of the non-deterministic commuter can be implemented by a local network without affecting the observable behaviour of the system. We build such internal infrastructure by means of simple components, called *connectors*, of the form $c(u_1u_2v)$. Connectors echo the information received from u_1 (call it the *input* node) over u_2 (the *output* node) using a *service* node v for the matching. Once v has served its purpose, a new service node is created. In symbols: $c(u_1u_2v) \xrightarrow{\substack{u_1,\bar{a},v \\ u_2,a,v}} \nu w\, c(u_1u_2w)$. The internal channel of the commuter in Figure 1 can be

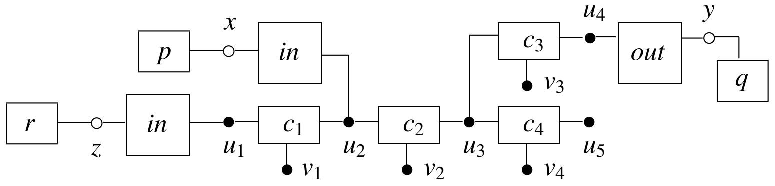

Figure 2. An implementation of the commuter in Figure 1 (we draw labeled boxes for hyperedges and bullets for nodes; the latter are solid when restricted and clear otherwise; tentacles are represented by lines connecting hyperedges with nodes).

implemented by the net G of four connectors in Figure 2. In symbols, grouping all indexed names into vectors:

$$G = \nu \boldsymbol{u}\,\boldsymbol{v}\; p(x) \,|\, q(y) \,|\, r(z) \,|\, in\,(z\,u_1) \,|\, in\,(x\,u_2) \\ |\, c_1(u_1u_2v_1) \,|\, c_2(u_2u_3v_2) \,|\, c_3(u_3u_4v_3) \,|\, c_4(u_3u_5v_4) \,|\, out\,(u_4y).$$

With this implementation, the transition in Example 1 is simulated by an equivalent transition $G \xrightarrow{\Lambda} H$, where (by ignoring all the unused sockets and connectors) $H = r(y) \,|\, q(y) \,|\; \nu \boldsymbol{u}\,\boldsymbol{w}\,(c_1(u_1u_2w_1) \,|\, c_2(u_2u_3w_2) \,|\, c_3(u_3u_4w_3) \,|\, out\,(u_4y))$ and Λ is $\{(u_1, a, z), (u_1, \overline{a}, v_1), (u_2, a, v_1), (u_2, \overline{a}, v_2), (u_3, a, v_2), (u_3, \overline{a}, v_3), (u_4, a, v_3), (u_4, \overline{a}, y)\}$.

In general, a graph made of sockets and connectors behaves like a non-deterministic commuter when it is a tree (that is, connected and acyclic) in which output sockets are attached by their first tentacle, input sockets by their second, no connector is attached by its service node, and moreover there exists a node, called *pivot*, that may split the graph into two (possibly disconnected) subgraphs, one including all the input and the other all the output sockets. In our implementation, nodes u_2, u_3 and u_4 are all pivotal. Of course, in the absence of a pivot, the internal infrastructure may allow for parallel connections, which are not contemplated in the specification of Example 1.

Proposition 1. *Any abstract graph* **G** *satisfying the conditions above is bisimulation equivalent to the abstract graph corresponding to the non-deterministic commuter obtained by deleting all the connectors from* **G** *and attaching all sockets to the pivot node.*

4.2 Functional equivalence

We now consider a more general kind of non-deterministic commuters, allowing multiple connections to occur at once. Hence, the internal structure of a commuter can now be any acyclic graph of connectors where all nodes are restricted. Input sockets are attached by their second tentacle, while their first is attached to a free node called *input node*; and dually for output sockets and *output nodes*.

A *connection* in a commuter C is a path from an input to an output node of C. A set of *disjoint* such connections (i.e. no node is shared by two connections in the set) is called a *service* of C. If s is a service, we write $\hat{s}$ the *partial* function from the input to the output nodes of C such that $\hat{s}(x) = y$ if and only if there exists a connection from x to y in s. We say that two commuters are *functionally equivalent* when, for each service s of one, there exists a service r of the other such that $\hat{s} = \hat{r}$, and vice-versa.

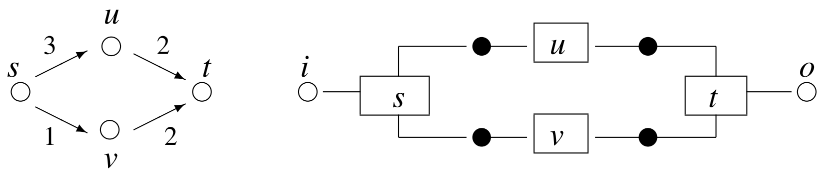

Figure 3. A channel with maximum flow 3 and its SG representation

Proposition 2. *Two non-deterministic commuters are functionally equivalent if and only if their alpha-equivalence classes are bisimulation equivalent.*

4.3 The *maximum flow* in a net

Consider an application where a *sender* sends discrete pieces of information, called *items*, to a *receiver*. The communication infrastructure is represented by a directed acyclic graph $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges. An edge (u, v) is an *input* for v and an *output* for u. We assume that the graph features a *unique* vertex with no input edges, called *source*, representing the sender. Similarly, the receiver is represented by a unique vertex with no input edges, called *target*.

We further endow each edge with a *capacity*, that is an upper bound to the number of items it can transmit *at a time*: n items can travel simultaneously through an edge provided its capacity is not less than n. No items are lost during transmission, and all items in input to a node are immediately presented in output in equal number. Hence, if the sender feeds the net with n items simultaneously, n items are received at once by the receiver, provided the edge capacities are not exceeded. This gives rise to the notion of *network flow* and of *maximum network flow* [6], i.e. the maximum number of items which can be simultaneously fed to the net. We call *channels*, and use metavariables A, $B \ldots$ to denote them, graphs $(\mathcal{V}, \mathcal{E})$ as above, endowed with a function $c : \mathcal{E} \to \mathbb{N}$ assigning to each edge an integer capacity, which we assume *strictly* greater than 0.

A *flow* in a channel A is a function $f : \mathcal{E} \to \mathbb{N}$ such that $f(u, v) \leq c(u, v)$ and $\sum_u f(u, v) = \sum_w f(v, w)$, for all $v \in \mathcal{V}$ except for source and target. The value of a flow f at the source s is $f(s) = \sum_u f(s, u)$, while $f(v) = \sum_u f(u, v)$ for all other vertices v. Clearly, $f(s) = f(t)$, and we call this number the *value* of f in A. A *positive* flow is one with value strictly greater than 0. The *maximum flow* of A, written $\phi(A)$, is the greatest value of a flow in A.

A channel $A = (\mathcal{V}, \mathcal{E}, c)$ is modelled by a synchronising graph $\hat{A}$ as follows. The nodes of $\hat{A}$ are the elements of $\mathcal{E} \cup \{i, o\}$, where i and o are called respectively the *input* and *output* nodes. All nodes in $\hat{A}$ are restricted, except i and o. Hyperedges are the vertices of $\mathcal{V}$. They are attached to nodes as follows: $v\,(\boldsymbol{x}\boldsymbol{y})$ represents a vertex $v \notin \{s, t\}$ where $\boldsymbol{x}$ is a vector including all input edges of v and $\boldsymbol{y}$ all outputs. Source and target are respectively $s\,(i\,\boldsymbol{y})$ and $t\,(\boldsymbol{x}\,o)$. Figure 3 represents a channel and its representation as a synchronising graph.

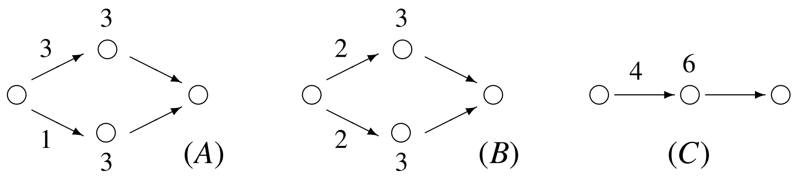

Figure 4. Three consumable channels. Omitted capacities and energy charges are assumed high enough as to not influence the flow dynamics.

The theory of channels features actions of the form n and $\overline{n}$, where $n \in \mathbb{N}$, and no parameters. It is generated by all axioms of the form:

$$v\,(x_1 \ldots x_n\, y_1 \ldots y_m) \xrightarrow[\;y_1\,\overline{k}_1,\ldots,y_m\,\overline{k}_m\;]{x_1\,h_1,\ldots,x_n\,h_n} v\,(x_1 \ldots x_n\, y_1 \ldots y_m),$$

where $\sum_{i=1}^{n} h_i = \sum_{j=1}^{m} k_j$ and, assuming $c(i) = c(o) = \infty$, $h_i \leq c(x_i)$, $k_j \leq c(y_j)$ for all nodes x_i and y_j. It is easy to see that A supports a flow of value k if and only if $\hat{A}$ has a transition Λ whose only observable actions are $\Lambda(i) = \{k\}$ and $\Lambda(o) = \{\overline{k}\}$. The following result shows that, in this simple model, bisimulation equivalence captures precisely the notion of maximum flow.

Proposition 3. *Let $\hat{A} \in \mathbf{A}$ and $\hat{B} \in \mathbf{B}$; then, $\mathbf{A} \sim \mathbf{B}$ if and only if $\phi\,(A) = \phi\,(B)$.*

4.4 The *dynamic flow* of a net

In real applications the nodes of a wireless network are often supplied with a *finite* amount of energy which is consumed in routing information. The *Smart Dust* [25], where nodes are *motes* of 1mm diameter, is an extreme example of energy-sensitive application. We give a simple account of such scenarios by charging the channels of Section 4.3 with consumable energy and studying their behaviour.

A *consumable channel* $A = (\mathcal{V}, \mathcal{E}, c, \eta)$ is a channel as above, endowed with an energy function $\eta : \mathcal{V} \to \mathbb{N}$. A flow in A is just as in Section 4.3, with the additional requirement that $f(v) \leq \eta\,(v)$ for all $v \in \mathcal{V}$. The energy inside a channel decreases at each flow. For simplicity, we shall assume that the passage of one information item through a vertex consumes one energy unit. Then, the energy dynamics is described by a transition system over consumable channels with transitions $(\mathcal{V}, \mathcal{E}, c, \eta) \xrightarrow{k} (\mathcal{V}, \mathcal{E}, c, \eta'\,)$, whenever the channel to the left admits a flow f of value k, and $\eta'(v) = \eta\,(v) - f(v)$ for all $v \in \mathcal{V}$. Clearly, $A \xrightarrow{k} A'$ implies $k \leq \phi\,(A)$.

A *computation* of a channel A is a sequence of transitions $A \xrightarrow{k_1} A_1 \ldots \xrightarrow{k_n} A_n$, which we shorten as $\langle k_1, \ldots, k_n \rangle$. The *dynamics* $\Phi\,(A)$ of a channel A is the set of all its computations. The channels depicted in Figure 4 all have a maximum flow of 4. However, while (B) and (C) have same dynamics, not so for (A) as it does not admit a computation $\langle 4, 2 \rangle$ while the others do. In Section 4.5 we distinguish channels such as (B) and (C) by introducing the notion of *robustness*.

As before, we model a consumable channel A by a synchronising graph $\hat{A}$, and relate the dynamics of the former with the observable behaviour of the latter. $\hat{A}$ is defined just as in Section 4.3 except that the hyperedge $v_n(\boldsymbol{x}\,\boldsymbol{y})$ corresponding to a vertex v is now labelled by the energy $n = \eta(v)$. The axioms $v_n(\boldsymbol{x}\,y) \xrightarrow{\cdots} v_{n'}(\boldsymbol{x}\,y)$ are as in Section 4.3 with the additional requirement that, writing p the value $\sum_{j=1}^{m} k_j$ of the transition, $p \leq n$ and $n' = n - p$.

Channel dynamics do not provide a good notion of behavioural equivalence for consumable channels. For example, consider the channels (B) and (C) in Figure 4: by always taking the upper path, after $\langle 2, 1 \rangle$ (B) becomes a net that cannot transmit three information items simultaneously (because of the capacity bound on its lower edge), whereas (C) can always perform $\langle 2, 1, 3 \rangle$. Bisimulation equivalence captures such differences in channel behaviours; moreover, it also yields a technique for proving that two channels have identical dynamics, i.e. the same set of traces:

Proposition 4. *Let $\hat{A} \in \mathbf{A}$ and $\hat{B} \in \mathbf{B}$; then, $\mathbf{A} \sim \mathbf{B}$ implies $\Phi(A) = \Phi(B)$.*

4.5 Network robustness

The channels (B) and (C) of Figure 4 have the same dynamics in the world described in Section 4.4 but not in a more realistic setting where vertices may *fail*. In such a case B is to be considered more *robust* than C. Robustness is ususally defined as the minimum number of faults that would block a net. Here we show a model where bisimulation equivalence captures precisely this notion of robustness.

Since the interplay between robustness and dynamic flow is subtle, we shall make the simplifying assumption that every node has infinite energy and every edge has capacity 1. Since flow values are not of interest here, we further assume that channels may pass at most one information item at a time. We let $r(A)$ denote the minimum number of nodes that must be removed to disconnect a channel A (i.e. source from target).

We represent the behaviour of a faulty channel A by augmenting the theory of synchronising graphs of Section 4.4 with new axioms for failure. As anticipated, all hyperedges of $\hat{A}$ are now labelled by ∞, and failure is represented by a sudden drop of $v_\infty(\boldsymbol{x}\,\boldsymbol{y})$ to $v_0(\boldsymbol{x}\,\boldsymbol{y})$. The axioms of flow are just as in Section 4.4 with the only difference that $c(i) = c(o) = 1$. For modeling failure, we introduce a new action $\dagger$ and the following axiom schemes, where we write v when the energy of the vertex (∞ or 0) is irrelevant:

$$v_\infty(\boldsymbol{x}\,\boldsymbol{y}) \xrightarrow{\substack{x_i,\dagger \\ y_j,\dagger}} v_0(\boldsymbol{x}\,\boldsymbol{y}) \qquad s(i\,\boldsymbol{y}) \xrightarrow{\substack{i,0 \\ y_j,\bar{\dagger}}} s(i\,\boldsymbol{y}) \qquad t(\boldsymbol{x}\,o) \xrightarrow{\substack{x_i,\bar{\dagger} \\ o,\bar{0}}} t(\boldsymbol{x}\,o)$$

$$v(\boldsymbol{x}\,\boldsymbol{y}) \xrightarrow{\substack{x_i,\bar{\dagger} \\ y_j,\dagger}} v(\boldsymbol{x}\,\boldsymbol{y}) \text{ for } v \notin \{s,t\} \qquad v(\boldsymbol{x}\,\boldsymbol{y}) \xrightarrow{\substack{x_i,\dagger \\ y_j,\bar{\dagger}}} v(\boldsymbol{x}\,\boldsymbol{y}) \text{ for } v \notin \{s,t\}$$

The first axiom accounts for the failure of v while the two axioms to the bottom are to transmit such an information respectively towards the source and the target. Notice that, in the first axiom, we can freely pick any $x_i \in \boldsymbol{x}$ and $y_j \in \boldsymbol{y}$ since every x_i and y_j lie in a path from s to t (because we work with connected graphs). By the remaining two axioms, source and target *hide* occurrences of $\dagger$ by issuing 0 on the free input and output nodes. Hence, failures are not *explicitly* observable. Note that we engineered our model as to admit one failure at a time. This allows us to test robustness by *counting* the number of steps that a channel requires in order to die. Simultaneous failures could of course have been modeled, at the cost of exposing the failure action † over the free nodes i and o.

The following result shows that, in this model, the robustness of a faulty channel is captured precisely by the notion of bisimulation equivalence.

Proposition 5. *Let* $\hat{A} \in \mathbf{A}$ *and* $\hat{B} \in \mathbf{B}$*; then,* $\mathbf{A} \sim \mathbf{B}$ *if and only if* $r(A) = r(B)$*.*

5 Conclusions

Synchronised graph rewriting has been proposed as a unifying semantic framework for process calculi [12,10,15,3]; to fulfill this project, graphs must be endowed with an abstract notion of behaviour. In this paper we do so by introducing a notion of bisimulation equivalence for a system of context-free synchronising graphs and by proving it a congruence with respect to parallel composition and node restriction. Bisimulation equivalence can be used to prove the correctness of system implementations, or (dually) of optimisation steps. For example, we have developed an application where the specification of a simple component, called non-deterministic commuter, is shown to be equivalent to an implementation in which the internal communication channel is replaced by a local net.

Bisimulation techniques could have been used, of course, *directly* in each one of the applications we have considered, without passing through an encoding into SG. However, the gain from our effort is twofold. On the one hand, matching the proposed notion of graph bisimulation with well known properties in the theory of networks is a good test for naturality and flexibility. On the other hand, SG may provide "mechanical" support for reasoning about such properties: systems such as the *Concurrency workbench* [5] support bisimulation proofs in the framework of process algebra. It is a challenging project to endow the current implementation of SG [23] with a similar capability.

Finally, our rule of synchronisation is reminiscent of the communication law of the *Fusion Calculus* [21]. Linking to Fusion is therefore a natural gateway for us to the universe of process algebra. We are working through this direction and have developed a context-free theory of synchronising graphs which can be viewed, in a precise sense, as a parallel and syntax-free version of the Fusion calculus. We believe that our translation is fully abstract w.r.t. proper notions of bisimulation equivalences, but we still have not been able to prove such a result.

Related work SG is closely related to the synchronised hyperedge rewriting (SHR) approach [10] from which it takes inspiration. SHR rewriting acts on *syntactic judgements*, that is *term-graphs* equipped with an *interface* consisting of their set of free nodes. The syntax-driven presentation of SHR enables several properties to be proven at a rather high abstraction level. For example, mimicking the approach in [20] for the π-calculus, it is proven in [14] that a given notion of bisimulation is a congruence for SHR parametrically in a synchronisation algebra with mobility, thus accounting for several styles of interaction. While renouncing the generality of SHR in abstracting over synchronisation algebras, SG exhibits a much simpler system of inference rules.

Productions are built-in SHR so as to make rewriting context-independent in a much similar way our productions do in context-free theories. Indeed, in SHR the dependencies described in Section 3 (and, in particular, those in Examples 4 and 5) are avoided by defining rewriting rules on productions rather than on graphs. Albeit being resolutive, this approach introduces some complexity in the definition of the operational semantics of SHR. For example, all possible instances must be considered when synchronising productions. We prefer to maintain the simple presentation of Section 2 and apply the (simpler) rewriting rules [sync], [open] and [res] to contex-free theories.

However, even in the setting of contex-free theories, SHR still differs from SG both in the notion of transition and in the proof theory. For example, while $x \vdash e(x) \xrightarrow{(x,a,y),\, id} x, y \vdash d(x, y)$ is legal in SHR, where nodes are treated as variables, $e(x) \xrightarrow{x,a,y} d(x, y)$ violates the principle of locality of Definition 1 in SG, where nodes are "constants". As for the proof theory, consider an application in which agent $d(x)$ dies. This is done in SG by the production $d(x) \xrightarrow{\emptyset} \emptyset$, which is mimicked in SHR by the production $x \vdash d(x) \xrightarrow{\emptyset,\, id} x \vdash nil$. Say this transition occurs in a larger context including an idle agent $e(y)$. In SG: $d(x) \,|\, e(y) \xrightarrow{\emptyset} e(y)$. In SHR: $x, y \vdash d(x) \,|\, e(y) \xrightarrow{\emptyset,\, id} x, y \vdash e(y)$. Node x remains in the context, even if no edge is attached to it. After that, and for the rest of its life, $e(y)$ can procede computation in SHR only if synchronising with the identical transition $x \vdash nil \xrightarrow{\emptyset,\, id} x \vdash nil$ of the graph consisting of a unique node x and no edges (no such a graph exists in SG). Identities are therefore fundamental in SHR, and all syntactic judgements are granted one. On the other hand, identities can be provided in a context-free theory *if desired*. Interleaving could be inhibited, for example, by not providing identities. This also impacts on behavioural equivalence as no distinction can be made in SHR between edges whose only transition is the identity and edges with no transitions at all.

The above examples show that no sensible matching can be made between SHR (with synchronisation à la Milner) and context-free theories of synchronising graphs, and none can be viewed as generalising the other.

Other interesting approaches have been applied to give congruential observational semantics to graph rewriting. Notably, in [9] *borrowed contexts* enable the derivation *minimal contexts* in a DPO (double-push out) approach. The idea, inspired by [17,16], consists in computing the minimal context within which a system can react. The resulting observational semantics, where observations are given by such minimal contexts, provides a bisimulation which is a congruence "by construction". A similar approach is taken in [18], where bigraphs are equipped with rules to form a *bigraphical reactive system* providing a bisimilarity which is a congruence. An interesting research direction is applying the mentioned approaches to SG and then compare the resulting observational semantics with the one defined here. Indeed, it is not clear what are the relationships between "natural" equivalences and those obtained via borrowed-context or reactive approaches. Initial studies for process algebras show that such equivalences may not coincide: for example, [11] shows a congruential bisimilarity obtained with a borrowed-context approach that is finer than open bisimilarity in the π-calculus. Recently, the notion of *saturated semantics* [1] has been shown to provide suitable congruential bisimilarities (e.g. the open bisimilarity for π-calculus can be obtained). This approach is quite promising but, at the best of our knowledge, it has not been applied to observational semantics of graphs rewriting.

References

1. F. Bonchi, B. König, and U. Montanari. Saturated semantics for reactive systems. In *Proc. of LICS*, pages 69–80, IEEE 2006.
2. L. Cardelli and A. Gordon. Mobile Ambients. *Theor. Comp. Science*, 1(240):177–213, 2000.
3. P. Cenciarelli, I. Talamo, and A. Tiberi. Ambient Graph Rewriting. *Electronic Notes in Theoretical Computer Science*, 117:335–351, 2005.
4. P. Cenciarelli and A. Tiberi. Rational Unification in 28 Characters. *Electronic Notes in Theoretical Computer Science*, 127-5:3–20, 2005.
5. R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench: A semantics-based tool for the verification of concurrent systems. *ACM ToPLaS*, 15(1):36–72, 1993.
6. T. Cormen, C. Leiserson and R. Rivest. *Introduction to algorithms*. MIT Press, 1990.
7. A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe. Algebraic Approaches to Graph Transformation I: Basic Concepts and Double Pushout Approach. In *Handbook of Graph Grammars and Computing by Graph Transformation*, vol. 1, chap.3.
8. P. Degano and U. Montanari. A model for distributed systems based on graph rewriting. *Journal of the ACM*, 34:411–449, 1987.
9. H. Ehrig and B. König. Deriving Bisimulation Congruences in the DPO Approach to Graph Rewriting. In *Proc. of FoSSaCS*, volume 2987 of *LNCS*, pages 151–166. Springer, 2004.
10. G. Ferrari, U. Montanari, and E. Tuosto. A LTS semantics of ambients via graph synchronization with mobility. In *Proc. of ICTCS'01*, volume 2202 of *LNCS*. Springer, 2001.
11. G. Ferrari, U. Montanari, and E. Tuosto. Model Checking for Nominal Calculi. In *Proc. of FoSSaCS*, volume 3441 of *LNCS*, pages 1–24. Springer, 2005.
12. D. Hirsch and U. Montanari. Synchronized hyperedge replacement with name mobility: A graphical calculus for name mobility. *Proc. of CONCUR*, volume 2154 of *LNCS*, 2001.
13. B. König and U. Montanari. Observational equivalence for synchronized graph rewriting with mobility. In *Proc. of TACS*, volume 2215 of *LNCS*, pages 145–164, 2001.
14. I. Lanese. *Synchronization Strategies for Global Computing Models*. PhD thesis, Univ. of Pisa, 2006.
15. I. Lanese and U. Montanari. A graphical fusion calculus. In *Proc. of COMETA'03*, 2003.
16. J. Leifer. *Operational Congruences for Reactive Systems*. PhD thesis, Univ. of Cambridge (UK), 2001.
17. J. Leifer and R. Milner. Deriving Bisimulation Congruences for Reactive Systems. In *Proc. of CONCUR*, volume 1877 of *LNCS*, pages 243–258, 2000.
18. R. Milner. Bigraphical Reactive Systems. *Proc. of CONCUR'01*, volume 2154 of *LNCS*.
19. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Information and Computation*, 100:1–77, 1992.
20. U. Montanari and M. Buscemi. A First Order Coalgebraic Model of π-Calculus Early Observational Equivalence. *Proc. of CONCUR'02*, volume 2421 of *LNCS*, pages 449–465.
21. J. Parrow and B. Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *Proc.of LICS*. IEEE Computer Society, 1998.
22. J. Riely and M. Hennessy. Distributed Processes and Location Failures. *Theoretical Computer Science*, 266:693–735, 2001.
23. I. Talamo. A Workbench for Distributed Programming. Masters thesis, http://briantb.unixcab.org/shetemp/, 2004.

24. D. Turi and G. D. Plotkin. Towards a mathematical operational semantics. In *Proc. of LICS*. IEEE Computer Society Press, 1997.
25. B. Warneke, M. Last, B. Liebowitz, and K. S. J. Pister. Smart dust: Communicating with a cubic-millimeter computer. *IEEE Computer*, 34(1):44–51, 2001.

Appendix

Proofs from Section 2

Proof of Lemma 1 Let $G \xrightarrow{\Lambda} H$ and $F \xrightarrow{\Theta} K$ be transitions, and let x belong to $fn(G)$. Any transition Ξ in $(\nu x\,\Lambda) * \Theta$ must be of the form $(\nu x\,G)|F \xrightarrow{\Lambda\cup\Theta} \rho(H'\,|K)$, where $H' = H$ if $x \in dng(\Lambda)$ and $H' = \nu x\,H$ otherwise. Note first that, since $\nu x\,\Lambda\#\Theta$ implies $\Lambda\#\Theta$, the transition $G|F \xrightarrow{\Lambda\cup\Theta} \rho(H|K)$ is in $\Lambda*\Theta$. Let $x \in dng(\Lambda)$. Since Ξ is a transition, $x \notin \rho(H|K)$ by condition 3 of definition 1. Then, independently of whether [open] or [res] is applied to $G|F \xrightarrow{\Lambda\cup\Theta} \rho(H|K)$, we obtain $(\nu x\,G)|F = \nu x\,(G|F) \xrightarrow{\Lambda\cup\Theta} \rho(H'\,|K)$ in $\nu x\,(\Lambda * \Theta)$ as required. Otherwise, let $x \notin dng(\Lambda)$. If $x \notin |H|$, then $H' = \nu x\,H = H$. If $x \in dng(\Lambda \cup \Theta)$ then Ξ is the result of applying [open] to $G|F \xrightarrow{\Lambda\cup\Theta} \rho(H|K)$. On the other hand, since Ξ satisfies condition 3, $x \notin dng(\Lambda \cup \Theta)$ implies $x \notin \rho(H|K)$, and hence Ξ is obtained again from $\Lambda \cup \Theta$ by [res]. Finally, if $x \in |H|$, the synchronisation of $\nu x\,G \xrightarrow{\Lambda} \nu x\,H$ with Θ has no effect on x. Hence $\rho(\nu x\,H|K) = \nu x\,\rho(H|K)$ and moreover, again by condition 3, $x \notin dng(\Lambda\cup\Theta)$. Then Ξ is the result of applying [res] to $(\Lambda\cup\Theta) \in \Lambda * \Theta$ as above. □

Proof of Lemma 2 We show one inclusion; the other is proven likewise. Let $F|G \xrightarrow{\Lambda\cup\Theta} \rho(H|K)$ be in $\Lambda * \Theta$ and let $F|G|I \xrightarrow{\Lambda\cup\Theta\cup\Xi} \sigma(\rho(H|K)|J)$ be in $(\Lambda * \Theta) * \Xi$. It is easy to check that $(\Lambda\cup\Theta)\,\#\,\Xi$ if and only if $\Lambda\,\#\,\Xi$ and $\Theta\,\#\,\Xi$. Hence, by synchronising Ξ with Θ, and the result with Λ, we obtain a transition $F|G|I \xrightarrow{\Lambda\cup\Theta\cup\Xi} \sigma(H|\pi(K|J))$ in $\Lambda\,\#\,(\Theta\,\#\,\Xi)$. The result follows by noticing that $\sigma\rho = \sigma\pi = \sigma$ and hence $\sigma(\rho(H|K)|J) = \sigma(H|K|J) = \sigma(H|\pi(K|J))$. □

Proofs from Section 3

Parallel and sequential composition have useful meta-theoretical properties in context-free theories. Let Λ be any transition of a composite graph $G|F$; there exists a Y-shaped derivation of Λ where the actions of G and those of F are synchronised separately in each branch of the Y. More precisely:

Theorem 2. *Let Λ be a transition in a context-free theory, and let $G|F$ be its source. Then, Λ is an element of a set $\nu\mathbf{x}\,(\Theta * \Xi)$, where G has exactly the same hyperedges as the source of Θ and F as the source of Ξ.*

Proof. Observe that, if x does not occur or it is restricted in the source of a transition Λ, then $\nu x\,\Lambda = \Lambda$. Hence, the hypothesis of Lemma 1 can be assumed to hold for any application of [open] and [res] in a derivation tree. All such applications can therefore be moved toward the root of the tree. Moreover, since the source of productions are single-hyperedged, all applications of [sync] can be reshuffled by Lemma 2 so as to separate the actions of G from those of F. □

Similarly, synchronisations occurring in parallel can be serialised, provided the axioms of the theory are suitably simple. We call *simple* a transition Λ such that $|dom(\Lambda)| \leq 1$. A simple transition of the form $G \xrightarrow{\emptyset} G$ is called an *identity*.

Theorem 3. *Let $G \xrightarrow{\Lambda \cup \Theta} H$ be a transition in a context-free theory generated from simple axioms including the identities, and let $\mathrm{dom}(\Lambda) \cap \mathrm{dom}(\Theta)$ be empty. Then $G \xrightarrow{\Lambda \cup \Theta} H$ factorises as $G \xrightarrow{\Lambda} F \xrightarrow{\rho\,\Theta} H$, where ρ is a unifier of Λ.*

Corollary 1. *Any transition in a context-free theory with simple axioms and identities factorises as a sequence of simple transitions.*

Lemma 3. *Let $h : \mathcal{N} \to \mathcal{N}$ be a node substitution and let $hG \xrightarrow{\Psi} H$ be derivable from a set of productions. Then, Ψ is of the form $hG \xrightarrow{h\Lambda} \nu \boldsymbol{x}\,\rho(hK)$, where $G \xrightarrow{\Lambda} K$ is a derivable transition and ρ is a unifier of $h\Lambda$.*

Proof. By induction on the depth of the derivation tree. If Ψ is an axiom, the statement holds with Λ obtained as an instance of the same production as Ψ. This is because each tentacle in the source of a production is attached to a distinct node. As for the inductive steps, we only show the case where Ψ is the result of a synchronisation. The others are similar.

Let Ψ be a transition $h(G_1|G_2) \xrightarrow{\Psi_1 \cup \Psi_2} \rho(H_1|H_2)$ obtained by synchronising $hG_1 \xrightarrow{\Psi_1} H_1$ and $hG_2 \xrightarrow{\Psi_2} H_2$. By inductive hypothesis Ψ_1 and Ψ_2 are respectively of the form $hG_1 \xrightarrow{h\Lambda_1} \nu \boldsymbol{x}_1 \rho_1(hK_1)$, and $hG_2 \xrightarrow{h\Lambda_2} \nu \boldsymbol{x}_2 \rho_2(hK_2)$, and the transitions $G_1 \xrightarrow{\Lambda_1} K_1$ and $G_2 \xrightarrow{\Lambda_2} K_2$ are derivable. Since ρ_1 unifies Ψ_1, while ρ unifies $\Psi_1 \cup \Psi_2$, we have $\rho \circ \rho_1 = \rho$, and similarly for ρ_2. Moreover, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are not affected by the synchronisation of Ψ_1 and Ψ_2, and hence by ρ. Therefore, writing $\boldsymbol{x}$ for $\boldsymbol{x}_1\boldsymbol{x}_2$, we have:

$$\rho(H_1|H_2) = \rho(\nu \boldsymbol{x}_1 \rho_1 hK_1 | \nu \boldsymbol{x}_2 \rho_2 hK_2) = \nu \boldsymbol{x}\, \rho(\rho_1 hK_1 | \rho_2 hK_2) = \nu \boldsymbol{x}\, \rho\, h(K_1|K_2).$$

Let $G_1|G_2 \xrightarrow{\Lambda_1 \cup \Lambda_2} \sigma(K_1|K_2)$ result from synchronising Λ_1 and Λ_2, and let x be a node of $K_1|K_2$. Since $\sigma x \in [x]_{\Lambda_1 \cup \Lambda_2}$, we have $h\sigma x \in h[x]_{\Lambda_1 \cup \Lambda_2} \subseteq [hx]_{h(\Lambda_1 \cup \Lambda_2)}$. Hence, $\rho\, h\sigma x = \rho\, hx$. Then, $\rho\, h\sigma(K_1|K_2) = \rho\, h(K_1|K_2)$ as required. □

We now move to the main result, i.e. that bisimulation equivalence is a congruence. First of all, we prove that it is closed under arbitrary node substitutions and then we use this result to prove closure under parallel compositions and restrictions. What we are going to prove is done for $\prec$, but the results also hold for $\sim$, since all the relations we are going to define are symmetric.

In the proofs, we shall use both abstract and ordinary graphs. However, when we work with ordinary graphs, it will be useful to relate names occurring in two equivalent transitions. Thus, if $G \xrightarrow{\Lambda} G'$ and $F \xrightarrow{\Theta} F'$ are equivalent transitions, we say that $x \in |\Lambda|$ and $y \in |\Theta|$ are *observationally related* if either $x = y \in fn(G) \cap fn(F)$ or there exists $o \in obs(\Lambda)$ such that $x \overset{\Lambda}{=}_o o \overset{\Theta}{=}_o y$. Note that the relation is symmetrical because $obs(\Lambda) = obs(\Theta)$ whenever Λ and Θ are equivalent. Note also that observational relation is not restricted to observable nodes, being it defined by $\overset{\Lambda}{=}_o$ and $\overset{\Theta}{=}_o$ rather then by $\overset{\Lambda}{\simeq}$ and $\overset{\Theta}{\simeq}$.

Lemma 4. *Simulation is preserved by node substitution:* $\mathbf{G} \prec \mathbf{F}$ *implies* $h\mathbf{G} \prec h\mathbf{F}$*, for all* $h : \mathcal{N} \to \mathcal{N}$.

Proof. Let

$$\mathcal{R} = \{(\nu\boldsymbol{x}\, h\mathbf{G}, \nu\boldsymbol{x}\, h\mathbf{F}) \mid \mathbf{G} \prec \mathbf{F} \text{ and } dom(h) \subseteq fn(\mathbf{G}) \cup fn(\mathbf{F})\}$$

Consider a pair $(\nu\boldsymbol{x}\, h\mathbf{G}, \nu\boldsymbol{x}\, h\mathbf{F}) \in \mathcal{R}$ and a transition $\nu\boldsymbol{x}\, h\mathbf{G} \xrightarrow{\Psi} \mathbf{H}$ fair with $\nu\boldsymbol{x}\, h\mathbf{F}$; by definition and Lemma 3, we have that $\nu\boldsymbol{x}\, hG \xrightarrow{h\Lambda} \nu\boldsymbol{u}\, \sigma hH$, where $G \in \mathbf{G}$, $h\Lambda = \Psi$, $\nu\boldsymbol{u}\, \sigma hH \in \mathbf{H}$, σ is a unifier of $h\Lambda$ and is the identity on $\boldsymbol{u}$, and $G \xrightarrow{\Lambda} H$. Since $\mathbf{G} \prec \mathbf{F}$ and Λ is fair for $\mathbf{F}$, there exists $F \in \mathbf{F}$ such that $F \xrightarrow{\Theta} \sigma' K$, Λ and Θ are equivalent, σ' is a unifier of Θ and $\mathbf{H} \prec \mathbf{K}$. This implies that $\nu\boldsymbol{x}\, hF \xrightarrow{h\Theta} \nu\boldsymbol{v}\, \sigma' hK$; moreover, $h\Lambda$ and $h\Theta$ are equivalent, σ' is a unifier of $h\Theta$ and it is the identity on $\boldsymbol{v}$. It remains to prove that the alpha-equivalence classes of $\nu\boldsymbol{u}\, \sigma hH$ and of $\nu\boldsymbol{v}\, \sigma' hK$ are in $\mathcal{R}$.

First, observe that σ (and σ') can be written as the composition of three substitutions, affecting different kinds of nodes: indeed, $\sigma = \sigma_f \circ \sigma_b \circ \sigma_{bf}$ where σ_f affects nodes free in $\nu\boldsymbol{x}\, hG$, σ_b nodes bound in both $\nu\boldsymbol{x}\, hG$ and in $\nu\boldsymbol{u}\, \sigma hH$ and σ_{bf} nodes bound in $\nu\boldsymbol{x}\, hG$ but free in $\nu\boldsymbol{u}\, \sigma hH$ (similarly for σ'). We can also assume σ_f and σ'_f be the same function, since $h\Lambda$ and $h\Theta$ are equivalent.

We now show that $\nu\boldsymbol{u}\, \sigma hH$ and $\nu\boldsymbol{v}\, \sigma' hK$ are alpha-equivalent to $\nu\boldsymbol{y}\, \sigma_f \bar{\sigma}_b \sigma_{bf} hH$ and $\nu\boldsymbol{y}\, \sigma_f \bar{\sigma}_b \sigma'_{bf} hK$, respectively. For every $u_i \in \boldsymbol{u}$, consider the class $[u_i]$ of the image under h of a node $y_i \in |G|$. If $y_i \in fn(G)$, because of the equivalence of Λ and Θ, there must be a class $[v_j]$ observationally related to $[u_i]$ containing $h(y_i)$; thus, we let $\bar{\sigma}_b(h[u_i]) = h(y_i)$ and $y_i \in \boldsymbol{y}$. On the other hand, if $[u_i]$ contains only nodes already bound in G, these nodes can not be affected by h; moreover in Θ there must be a class $[v_j]$ that is observationally related to $[u_i]$. In this case, choose a representative z and define $\bar{\sigma}_b([u_i]) = \bar{\sigma}_b([v_j]) = z$.

Next, we define a function $\bar{\sigma}_{bf}$ such that $\sigma_{bf} \circ h = \bar{\sigma}_{bf} = \sigma'_{bf} \circ h$. Observe that for each class $[x_i]$ that is affected by σ_{bf}, there is an observationally related class $[z_j]$ affected by σ'_{bf}. Consider $[x_i]$ containing the image under h of a node y free in G and let $[z_j]$ be the class observationally related to it. Notice that $[z_j]$ must contain $h(y)$. Define $\bar{\sigma}_{bf}([x_i]) = \bar{\sigma}_{bf}([z_j]) = h(y)$. Consider now the classes $[x_i]$ and $[z_i]$ which are observationally related and contain no free node of G (and hence of F). The fusion h must be the identity on these classes. The fusion $\bar{\sigma}_{bf}$ defined above can be extended so that on these classes $\sigma_{bf} = \bar{\sigma}_{bf}$ and $\sigma'_{bf} = \bar{\sigma}_{bf}$. Thus, $\nu\boldsymbol{y}\, \sigma_f \bar{\sigma}_b \sigma_{bf} hH$ and $\nu\boldsymbol{y}\, \sigma_f \bar{\sigma}_b \sigma'_{bf} hK$ (that are alpha-equivalent to $\nu\boldsymbol{u}\, \sigma hH$ and $\nu\boldsymbol{v}\, \sigma' hK$, respectively) are equal to $\nu\boldsymbol{y}\, (\sigma_f \circ \bar{\sigma}_b \circ \bar{\sigma}_{bf})H$ and $\nu\boldsymbol{y}\, (\sigma_f \circ \bar{\sigma}_b \circ \bar{\sigma}_{bf})K$, respectively; thus, by definition, their alpha-equivalence classes are in $\mathcal{R}$. This concludes the proof. □

Notice that the lemma may fail for non-context-free theories, as for instance that of Example 3. Note also that the lemma would have to be rephrased were we adopting the brute-force notion of node substitution of section 2.

Proposition 6. *Simulation is transitive:* $\mathbf{G} \prec \mathbf{F}$ *and* $\mathbf{F} \prec \mathbf{I}$ *imply that* $\mathbf{G} \prec \mathbf{I}$.

Proof. Let

$$\mathcal{R} = \{(\mathbf{G}, \mathbf{I}) \ : \ \exists \mathbf{F} \ s.t. \ \mathbf{G} \prec \mathbf{F} \wedge \mathbf{F} \prec \mathbf{I}\}$$

Let $(\mathbf{G}, \mathbf{I}) \in \mathcal{R}$ and $\mathbf{G} \xrightarrow{\Lambda} \mathbf{H}$ be a transition fair with $\mathbf{I}$, with $G \xrightarrow{\Lambda} H$, $G \in \mathbf{G}$ and $H \in \mathbf{H}$. Let $g : res(G) \to \mathcal{N}$ be an injective renaming of the bound names of G with some fresh names such that $\mathbf{G} \xrightarrow{g\Lambda} g\mathbf{H}$ is fair with $\mathbf{F}$. By hypothesis, there exists $\mathbf{F} \xrightarrow{\Theta} \mathbf{K}$ such that $g\Lambda$ and Θ are equivalent; moreover, $g\mathbf{H} \prec \mathbf{K}$. Let $F \xrightarrow{\Theta} K$, where $F \in \mathbf{F}$ and $K \in \mathbf{K}$. Let $f : res(F) \to \mathcal{N}$ be an injective renaming of the bound names of F with some fresh names such that $\mathbf{F} \xrightarrow{f\Theta} f\mathbf{K}$ is fair with $\mathbf{I}$. Again by hypothesis, there exists $\mathbf{I} \xrightarrow{\Xi} \mathbf{J}$ such that $f\Theta$ and Ξ are equivalent; moreover $f\mathbf{K} \prec \mathbf{J}$.

By Lemma 4, $(g^{-1}f^{-1})f\mathbf{K} \prec (g^{-1}f^{-1})\mathbf{J}$ and $g^{-1}g\mathbf{H} \prec g^{-1}\mathbf{K}$; since f and g are injective, the previous relations reduce to $g^{-1}\mathbf{K} \prec g^{-1}f^{-1}\mathbf{J}$ and $\mathbf{H} \prec g^{-1}\mathbf{K}$. Thus, by construction, it holds that $(\mathbf{H}, g^{-1}f^{-1}\mathbf{J}) \in \mathcal{R}$. Moreover, since the images of f and g were fresh, $g^{-1}f^{-1}\mathbf{I} = \mathbf{I}$; thus, $\mathbf{I} \xrightarrow{g^{-1}f^{-1}\Xi} g^{-1}f^{-1}\mathbf{J}$. If we prove that Λ and $g^{-1}f^{-1}\Xi$ (that, from now on, will be denoted by Ξ') are equivalent, we have done.

- First, we have to prove that $\overset{g\Lambda}{\simeq} = \overset{\Theta}{\simeq}$ and $\overset{f\Theta}{\simeq} = \overset{\Xi}{\simeq}$ imply that $\overset{\Lambda}{\simeq} = \overset{\Xi'}{\simeq}$. By expanding the definition of $\overset{\Lambda}{\simeq}$, we have that $(p, q) \in \overset{\Lambda}{\simeq}$ if and only if
  - either $p = (y, b, j)$, for some $y \in fn(\mathbf{G})$ such that $\Lambda y = \{(b\, z)\}$, and $q = z_j \in fn(\mathbf{G})$;
  - or there exists $x \in |G|$ such that $\Lambda x = \{(a\, \mathbf{y}), (\bar{a}\, \mathbf{z})\}$, with $p = y_i \in fn(\mathbf{G})$, $q = z_i \in fn(\mathbf{G})$, $p \in dom(\Lambda)$ or $p \overset{\Lambda}{=} n \in fn(\mathbf{G}) \setminus \{p\}$, and $q \in dom(\Lambda)$ or $q \overset{\Lambda}{=} m \in fn(\mathbf{G}) \setminus \{q\}$.

  Thus, in all cases p and q only contain (or are) nodes that are free in $\mathbf{G}$. Since $dom(g) \subseteq res(G)$, it holds that $(p, q) \in \overset{\Lambda}{\simeq}$ implies that $(p, q) \in \overset{g\Lambda}{\simeq}$; conversely, if $(p, q) \in \overset{g\Lambda}{\simeq}$, since the image of g is fresh (and, thus, $dom(g^{-1}) \cap fn(\mathbf{G}) = \emptyset$), we have that $(p, q) \in \overset{\Lambda}{\simeq}$. Thus, $(p, q) \in \overset{\Lambda}{\simeq}$ if and only if $(p, q) \in \overset{g\Lambda}{\simeq}$ that, by hypothesis, holds if and only if $(p, q) \in \overset{\Theta}{\simeq}$. With a similar reasoning, we have that $(p, q) \in \overset{\Theta}{\simeq}$ if and only if $(p, q) \in \overset{f\Theta}{\simeq}$ that, by hypothesis, holds if and only if $(p, q) \in \overset{\Xi}{\simeq}$. Again, the images of g and of f are fresh (and, thus, $(dom(g^{-1}) \cup dom(f^{-1})) \cap fn(\mathbf{I}) = \emptyset$); moreover, p and q only contain (or are) nodes that are free in $\mathbf{I}$. Thus, $(p, q) \in \overset{\Xi}{\simeq}$ if and only if $(p, q) \in \overset{\Xi'}{\simeq}$, as desired.
- Second, we have to prove that $\{x \in fn(\mathbf{G}) \ : \ |\Lambda x| = 2\} = \{z \in fn(\mathbf{I}) \ : \ |\Xi' z| = 2\}$, by knowing that $\{x \in fn(\mathbf{G}) \ : \ |g\Lambda x| = 2\} = \{y \in fn(\mathbf{F}) \ : \ |\Theta y| = 2\}$ and $\{y \in fn(\mathbf{F}) \ : \ |f\Theta y| = 2\} = \{z \in fn(\mathbf{I}) \ : \ |\Xi z| = 2\}$. Notice that g and f only work on the bound names of G and F; thus, $|g\Lambda x| = |\Lambda x|$ and $|f\Theta y| = |\Theta y|$, for every $x \in fn(\mathbf{G})$ and $y \in fn(\mathbf{F})$. Since the images of f and g were fresh, $dom(g^{-1}f^{-1}) \cap fn(\mathbf{I}) = \emptyset$; thus, $|\Xi' z| = |\Xi z|$ for every $z \in fn(\mathbf{I})$, and this suffices to conclude. □

Proof of Theorem 1 We show that the relation

$$\mathcal{R} = \{(\nu \boldsymbol{x}\,(\mathbf{G}|\mathbf{U}), \nu \boldsymbol{x}\,(\mathbf{F}|\mathbf{U}))\ :\ \mathbf{G} \prec \mathbf{F}\}$$

is a simulation. Let $\mathbf{G} \prec \mathbf{F}$. For simplicity we consider the transitions of $\mathbf{G}|\mathbf{U}$ and $\mathbf{F}|\mathbf{U}$, rather than $\nu\boldsymbol{x}\,(\mathbf{G}|\mathbf{U})$ and $\nu\boldsymbol{x}\,(\mathbf{F}|\mathbf{U})$; the general proof is only slightly more complicated. So, let $\mathbf{G}|\mathbf{U} \xrightarrow{\Phi} \mathbf{W}$ be a transition; thus, there exist $G \in \mathbf{G}$, $U \in \mathbf{U}$ and $W \in \mathbf{W}$ such that $G|U \xrightarrow{\Phi} W$.

We start by exhibiting an equivalent transition of $\mathbf{F}|\mathbf{U}$ to some $\mathbf{Z}$. To this aim, define the *focus* in a derivation of a transition Ξ to be Ξ itself if it is an axiom or the conclusion of a [sync]; otherwise the focus is that of the sub-derivation of the premise of Ξ. By Theorem 2, there exists a derivation of Φ in which all the actions of G are separated from those of U. Let $G_0|U_0 \xrightarrow{\Lambda\cup\Psi} \rho(H_0 \,|\, V_0)$ be its focus, with $G_0 \xrightarrow{\Lambda} H_0$ and $U_0 \xrightarrow{\Psi} V_0$ as premises, where $G = \nu\boldsymbol{x_0}\,G_0$ and $U = \nu\boldsymbol{u_0}\,U_0$. Since, by assumption, $\boldsymbol{x_0} \cap |U| = \emptyset$, there are no unsynchronised actions on $\boldsymbol{x_0}$ in Λ (such actions would otherwise occur unsynchronised in Φ as well), and a transition $G \xrightarrow{\Lambda} H$ can therefore be derived from $G_0 \xrightarrow{\Lambda} H_0$. Since $\mathbf{G} \prec \mathbf{F}$, there exist $F \in \mathbf{F}$ (that, without loss of generality, we can assume be such that $F|U$ is defined) and a transition $F \xrightarrow{\Theta} K$ which is equivalent to Λ and such that $\mathbf{H} \prec \mathbf{K}$, where $\mathbf{H}$ and $\mathbf{K}$ are the alpha-equivalence classes of H and K respectively. By Lemmata 1 and 2, there exists a derivation of Θ where all synchronisations are applied first. Let $F_0 \xrightarrow{\Theta} H_0$ be its focus, with $F = \nu\boldsymbol{y_0}\,F_0$, and let $F_0|U_0 \xrightarrow{\Theta\cup\Psi} \sigma(K_0 \,|\, V_0)$ be derived by synchronising $U_0 \xrightarrow{\Psi} V_0$ with $F_0 \xrightarrow{\Theta} H_0$. By restricting $\Theta \cup \Psi$ on $\boldsymbol{y_0}$ and $\boldsymbol{u_0}$, a transition $F|U \xrightarrow{\Theta\cup\Psi} Z$ is obtained, which is easily shown to be equivalent to Φ. Thus, the desired transition is $\mathbf{F}|\mathbf{U} \xrightarrow{\Theta\cup\Psi} \mathbf{Z}$, where $\mathbf{Z}$ is the alpha-equivalence class of Z.

We now have to prove that $(\mathbf{W}, \mathbf{Z}) \in \mathcal{R}$. Let $H = \nu\boldsymbol{x_1}\,H_0$ and $V = \nu\boldsymbol{w_1}\,V_0$, with $\boldsymbol{x_1} \subseteq \boldsymbol{x_0}$ and $\boldsymbol{w_1} \subseteq \boldsymbol{u_0}$. Since the nodes in $\boldsymbol{x_1}$ and $\boldsymbol{w_1}$ are not affected by the synchronisation of Λ and Ψ, $\nu\boldsymbol{x_1}\,\rho H_0 = \rho H$ and $\nu\boldsymbol{w_1}\,\rho V_0 = \rho V$; and similarly for σ. Hence, W and Z are respectively of the form $\nu\boldsymbol{x_2}\,\rho(H|V)$ and $\nu\boldsymbol{y_2}\,\sigma(K|V)$, where $\boldsymbol{x_2} \subseteq \boldsymbol{x_0}\boldsymbol{u_0}$ are the dangling nodes of Λ and Ψ which are fused by the synchronisation, and similarly for $\boldsymbol{y_2} \subseteq \boldsymbol{y_0}\boldsymbol{u_0}$. Summarising, we have to prove that the alpha-equivalence classes of $\nu\boldsymbol{x_2}\,\rho(H|V)$ and $\nu\boldsymbol{y_2}\,\sigma(K|V))$ are in $\mathcal{R}$.

Calling *interface* of a transition Ξ the set of its unsynchronised parameters $|\Xi|_\iota = \{(x, a, i) \in \mathcal{P}\ :\ \Xi x = \{(a, \boldsymbol{y})\}$ and $0 \le i \le |\boldsymbol{y}|\}$, we let $x R y$ hold on $|\Lambda \cup \Psi| \times |\Theta \cup \Psi|$ precisely when x and y are related either observationally or by the interface of Ψ, that is: $x \overset{\Lambda\cup\Psi}{=} o \overset{\Theta\cup\Psi}{=} y$, for some $o \in |\Psi|_\iota$. The projections of R form a pushout diagram

$$\begin{tikzcd}
R \arrow[r] \arrow[d] & {|\Theta \cup \Psi|} \arrow[d, "f"] \\
{|\Lambda \cup \Psi|} \arrow[r, "g"'] \arrow[ur, phantom, "\ulcorner", very near end] & N \subseteq \mathcal{N}
\end{tikzcd}$$

in which N is a set of *fresh* nodes, and g and f are such that $g\,x = f\,y$ if and only if $x R y$. Let $\xi\ :\ fn(H) \cup fn(K) \to \mathcal{N}$ be the node renaming function mapping x to

$g(x)$ if $x \in fn(H)$, or else to $f(x)$ if $x \in fn(K)$. This is a good definition because, if $x \in fn(H) \cup fn(K)$, then $g(x) = f(x)$. Since $fn(H) = fn(\mathbf{H})$ and $fn(K) = fn(\mathbf{K})$, it holds that $g(\rho\mathbf{H}) = \xi(\mathbf{H})$ and $f(\sigma\mathbf{K}) = \xi(\mathbf{K})$; thus, since $\mathbf{H} \prec \mathbf{K}$, Lemma 4 implies that

$$g(\rho\mathbf{H}) \prec f(\sigma\mathbf{K}). \tag{1}$$

Now, let x be a free node of V. Either x is free in U, in which case x (as an object of $|\Lambda \cup \Psi|$) is observationally related with itself (as an object of $|\Theta \cup \Psi|$) or there exists a parameter o in the interface of Ψ such that $x \overset{\Lambda\cup\Psi}{=} o \overset{\Theta\cup\Psi}{=} x$. In both cases $\rho x \, R \, \sigma x$ and, therefore, $g(\rho V) = f(\sigma V)$; this trivially implies that

$$g(\rho\mathbf{V}) = f(\sigma\mathbf{V}). \tag{2}$$

Moreover, there exists a vector $\boldsymbol{v}$ such that

$$\nu\boldsymbol{v}\, g\rho(H|V) = \nu(g\boldsymbol{x_2})\, g\rho(H|V) \text{ and} \tag{3}$$

$$\nu\boldsymbol{v}\, f\sigma(K|V) = \nu(f\boldsymbol{y_2})\, f\sigma(K|V). \tag{4}$$

In fact, any node $x \in \boldsymbol{x_2}$ must be either dangling in Λ or in Ψ. Since Λ and Θ are equivalent, there must exist $y \in |\Theta \cup \Psi|$ such that $x \overset{\Lambda\cup\Psi}{=} o \overset{\Theta\cup\Psi}{=} y$ for some $o \in |\Psi|_\iota$; hence, $x \, R \, y$ holds. Then, either $[y]_{\Theta\cup\Psi} \cap |\sigma(K|V)| = \emptyset$, in which case $gx \notin |f\sigma(K|V)|$, or else there exists $y' \in \boldsymbol{y_2}$ such that $gx = fy'$. In either cases $\nu(gx)\, \nu(f\boldsymbol{y_2}) f\sigma(K|V) = \nu(f\boldsymbol{y_2})\, f\sigma(K|V)$. The dual argument applies when $x \in \boldsymbol{y_2}$. So, we obtain the equations (3) and (4) by taking $\boldsymbol{v}$ to be $g\boldsymbol{x_2} \cup f\boldsymbol{y_2}$.

Now notice that $\nu\boldsymbol{x_2}\, \rho(H|V)$ and $\nu(g\boldsymbol{x_2})\, g\rho(H|V)$ are alpha-equivalent (because N is a set of fresh names), and so are $\nu\boldsymbol{y_2}\, \sigma(K|V)$ and $\nu(f\boldsymbol{y_2})\, f\sigma(K|V)$. Moreover, $\nu\boldsymbol{v}\, g\rho(\mathbf{H}|\mathbf{V})$ is the alpha-equivalence class of $\nu\boldsymbol{v}\, g\rho(H|V)$ and it is the same abstract graph as $\nu\boldsymbol{v}\, (g\rho\mathbf{H}|g\rho\mathbf{V})$; similarly, $\nu\boldsymbol{v}\, f\sigma(\mathbf{K}|\mathbf{V}) = \nu\boldsymbol{v}\, (f\sigma\mathbf{K}|f\sigma\mathbf{V})$ and it is the alpha-equivalence class of $\nu\boldsymbol{v}\, f\sigma(K|V)$. Noted these facts, we easily conclude:

$$\begin{array}{ll} \nu\boldsymbol{x_2}\, \rho(H|V) \overset{\alpha}{=} \nu(g\boldsymbol{x_2})\, g\rho(H|V) = \nu\boldsymbol{v}\, g\rho(H|V) & \text{by (3)} \\ (\nu\boldsymbol{v}\, (g\rho\mathbf{H}|g\rho\mathbf{V}),\ \nu\boldsymbol{v}(f\sigma\mathbf{K}|f\sigma\mathbf{V})) \in \mathcal{R} & \text{by (1), (2) and definition of } \mathcal{R} \\ \nu\boldsymbol{v} f\sigma(K|V) = \nu(f\boldsymbol{y_2})\, f\sigma(K|V) \overset{\alpha}{=} \nu\boldsymbol{y_2}\, \sigma(K|V) & \text{by (4).} \end{array}$$

□

Proofs from Section 4.1

Proof of Proposition 1. Writing $\mathbf{C}_G$ the commuter obtained from an abstract graph $\mathbf{G}$ of sockets and connectors as described in the hypothesis, the set of all pairs of the form $(\mathbf{G}, \mathbf{C}_G)$ is a bisimulation. In fact, any transition of $\mathbf{C}_G$ must involve the synchronisation of one (because the infrastructure is acyclic) and only one (because a pivot exists) input/output pair of sockets. By an easy check, the echoing actions $(u_1, \overline{a}, v)$ and (u_2, a, v) performed by the connectors preseve the equivalence of transitions. □

Proofs from Section 4.2

Lemma 5. *The services of a non-deterministic commuter are in one-to-one correspondence with its transitions. Moreover, if s corresponds to* Λ, *then* $\hat{s} = \overset{\Lambda}{\simeq}$.

Lemma 6. *Let C be a non-deterministic commuter and let $C \xrightarrow{\Lambda} D$ be a transition; s is a service of D if and only if $s = s_C - s_\Lambda$, where s_C is a service of C and s_Λ corresponds to Λ as by Lemma 5.*

Proof of Proposition 2. (*If*) Let $\mathbf{C}$ and $\mathbf{D}$ be bisimulation equivalent non-deterministic commuters; let $C \in \mathbf{C}$ and s be a service of C. By Lemma 5, C has a transition Λ with $\hat{s} = \overset{\Lambda}{\simeq}$. Since $\mathbf{C} \sim \mathbf{D}$, there exists $D \in \mathbf{D}$ and a transition Θ of D which is equivalent to Λ, and hence, again by Lemma 5, a service r of D such that $\hat{r} = \overset{\Theta}{\simeq} = \overset{\Lambda}{\simeq} = \hat{s}$.

(*Only if*) We show that functional equivalence extended to abstract graphs is a bisimulation. Let $\mathbf{C}$ and $\mathbf{D}$ be functionally equivalent, and let $\mathbf{C} \xrightarrow{\Lambda} \mathbf{E}$ be a transition fair with $\mathbf{D}$. Then, $\overset{\Lambda}{\simeq} = \hat{s}_\Lambda = \hat{r}_\Theta = \overset{\Theta}{\simeq}$, where $\mathbf{D} \xrightarrow{\Theta} \mathbf{F}$ is a transition and s_Λ and r_Θ are related respectively with Λ and Θ as in Lemma 5. If s is a service of (a representative of) $\mathbf{E}$, then $s = s_C - s_\Lambda$ by Lemma 6, where s_C is a service of (a representative of) $\mathbf{C}$. Since $\mathbf{C}$ and $\mathbf{D}$ are functionally equivalent, there exists a service r_D of (a representative of) $\mathbf{D}$ such that $\hat{s}_C = \hat{r}_D$. Then, again by Lemma 6, $r = r_D - r_\Theta$ is a service of (a representative of) $\mathbf{F}$ and $\hat{r} = \hat{r}_D - \hat{r}_\Theta = \hat{s}_C - \hat{s}_\Lambda = \hat{s}$. Since the argument is symmetrical, $\mathbf{E}$ is functionally equivalent to $\mathbf{F}$, as required. □

Proofs from Section 4.3

In what follows, we shall use notation $\Lambda|_v$ to denote $\{(e, \cdot) \in \Lambda \ : \ e = (u, v) \vee e = (v, u)\}$. Moreover, $in(v)$ and $out(v)$ will denote the input and output edges of a vertex v.

Lemma 7.

1. *If $\hat{A} \xrightarrow{\Lambda} F$ then there exists a $k \le \phi(A)$ s.t. $\Lambda(i) = \{k\}$, $\Lambda(o) = \{\bar{k}\}$, i and o are the only visible nodes in $\mathsf{dom}(\Lambda)$, $obj(\Lambda) = \emptyset$ and $F = \hat{A}$.*
2. *Conversely, for every $k \le \phi(A)$, there exists a Λ such that $\Lambda(i) = \{k\}$, $\Lambda(o) = \{\bar{k}\}$, i and o are the only visible nodes in $\mathsf{dom}(\Lambda)$, $obj(\Lambda) = \emptyset$ and $\hat{A} \xrightarrow{\Lambda} \hat{A}$.*

Proof. For the first part, notice that the transitions of a graph $\hat{\cdot}$ do not change the graph and have no objects; thus, trivially, $F = \hat{A}$ and $obj(\Lambda) = \emptyset$. Then, notice that the only visible nodes in $\hat{A}$ are i and o, so they are the only possible visible nodes in $\mathsf{dom}(\Lambda)$; moreover, since Λ is a transition, by definition every restricted node contained in $\mathsf{dom}(\Lambda)$ must host a synchronisation and, since every hyperedge incides at least one restricted node, i and o must occur in $\mathsf{dom}(\Lambda)$ and it must be $|\Lambda(i)| = |\Lambda(o)| = 1$. It remains to prove that $\Lambda(i) = \{k\}$ and $\Lambda(o) = \{\bar{k}\}$, for some $k \le \phi(A)$. To this aim, consider function f such that

$$f(u, v) = \begin{cases} h & \text{if } h \in \Lambda(e), \text{ for } e = (u, v) \\ 0 & \text{otherwise} \end{cases}$$

By definition of the transitions in $\hat{A}$, it follows that f is a flow in A. Thus,

$$\phi(A) \ge |f| = \sum_{e \in out(s)} f(e) = \sum_{e \in in(t)} f(e) \tag{5}$$

where the last equality easily follows from the definition of the value of a flow. Now, $N_s \xrightarrow{\Lambda|_s} N_s$, where $\Lambda|_s = \{(i,h)\} \cup \{(e,h_e) \ : \ e \in out(s)\}$ and

$$h = \sum_{e \in out(s)} h_e = \sum_{e \in out(s)} f(e) \tag{6}$$

Similarly, $N_t \xrightarrow{\Lambda|_t} N_t$, where $\Lambda|_t = \{(o,\bar{k})\} \cup \{(e,k_e) \ : \ e \in in(t)\}$ and

$$k = \sum_{e \in in(t)} k_e = \sum_{e \in in(t)} f(e) \tag{7}$$

The thesis follows from (5), (6) and (7).

For the second part, we can find a flow f with value k; we aim at proving that the desired transition is $\Lambda = \{(i,|f|),(o,\overline{|f|})\} \cup \{(e,f(e)),(e,\overline{f(e)}) \ : \ e \in \mathcal{E}\}$. The only non-trivial thing to prove is that $\hat{A} \xrightarrow{\Lambda}$. To this aim, first notice that $N_v \xrightarrow{\Lambda|_v} N_v$, where $dom(\Lambda|_v) = in(v) \cup out(v)$; indeed, since f is a flow, $\sum_{e \in out(v)} f(e) = \sum_{e \in in(v)} f(e)$ and $f(e) \leq c(e)$ for every $e \in in(v) \cup out(v)$. Then, notice that, if $u \neq v$, we have that either $e \notin dom(\Lambda|_u) \cap dom(\Lambda|_v)$ or $\Lambda|_u(e) = \{f(e)\}$ and $\Lambda|_v(e) = \{\overline{f(e)}\}$ (or vice versa); hence, $N_u \mid N_v \xrightarrow{\Lambda|_u \cup \Lambda|_v} N_u \mid N_v$. We can easily conclude by noting that $\Lambda = \bigcup_{v \in \mathcal{V}} \Lambda|_v$. □

Proof of Proposition 3. First, notice that, for every A, it holds that $\mathbf{A} \xrightarrow{\Lambda} \mathbf{G}$ if and only if $\hat{A} \xrightarrow{\Lambda} G$, for some $G \in \mathbf{G}$; thus, by definition of $\hat{A}$, every transition of $\mathbf{A}$ is also a transition of $\hat{A}$. Thus, in the bisimulation game, it suffices to work on the transitions of $\hat{A}$ (that coincides with those of $\mathbf{A}$).

For the 'if' part, we show that

$$\mathcal{S} = \{(\mathbf{A},\mathbf{B}) : \phi(A) = \phi(B) \wedge \hat{A} \in \mathbf{A} \wedge \hat{B} \in \mathbf{B}\}$$

is a simulation. Let $\hat{A} \xrightarrow{\Lambda} F$; by Lemma 7(1), $\Lambda(i) = \{k\}$, $\Lambda(o) = \{\bar{k}\}$, $k \leq \phi(A)$, i and o are the only visible nodes in $dom(\Lambda)$, $obj(\Lambda) = \emptyset$ and $F = \hat{A}$. Since $\phi(A) = \phi(B)$ and because of Lemma 7(2), $\hat{B} \xrightarrow{\Lambda'} \hat{B}$, for some Λ' such that $\Lambda'(i) = \{k\}$, $\Lambda'(o) = \{\bar{k}\}$, i and o are the only visible nodes in $dom(\Lambda')$ and with $obj(\Lambda') = \emptyset$; this suffices to conclude that Λ' is equivalent to Λ.

For the 'only if' part, let $k = \phi(A)$; by Lemma 7(2), we know that $\hat{A} \xrightarrow{\Lambda} \hat{A}$, for some Λ such that $\Lambda(i) = \{k\}$, $\Lambda(o) = \{\bar{k}\}$, i and o are the only visible nodes in $dom(\Lambda)$ and with $obj(\Lambda) = \emptyset$. By definition of bisimilarity, $\hat{B} \xrightarrow{\Lambda'} F$, for some Λ' equivalent to Λ. By Lemma 7(1), we can conclude that $k \leq \phi(B)$; but it cannot be $k < \phi(B)$, otherwise, to be bisimilar to $\hat{B}$, $\hat{A}$ would have to perform a Λ such that $\Lambda(i) = \{\phi(B)\}$ and $\Lambda(o) = \{\overline{\phi(B)}\}$, in contradiction with Lemma 7(1). □

Proofs from Section 4.4

Proofs for this refined scenario can be easily derived from those given in the previous subsection; we only give the statements.

Lemma 8. *If $A \xrightarrow{k} B$ is a transition of a consumable channel A, then the synchronising graph $\hat{A}$ has a transition $\hat{A} \xrightarrow{k} \hat{B}$. Conversely, if $\hat{A} \xrightarrow{k} F$ then there exists $A \xrightarrow{k} B$ such that $F = \hat{B}$.*

Proof. Similar to the proof of Lemma 7.

Proof of Proposition 4. Similar to the 'only if' part of Proposition 3.

Proofs from Section 4.5

To prove the correspondence between bisimilarity and robustness, we first need the notion of cut in a synchronised graph F of the theory defined by the previous axiom shemata. A *cut* for F is a subset of hyperedges C such that, for every positive flow f, there exists a $e(z) \in C$ such that $f(z_j) > 0$, for some j. The robustness of F, denoted $R(F)$, is the minimal cardinality of a cut in F.

Lemma 9. *Let F be a synchronised graph of the theory defined by the axiom shemata in Section 4.5. Then, $F \xrightarrow{\Lambda} F'$ implies that Λ and $R(F')$ can only be such that:*

1. *$\Lambda i = \{1\}$ and $R(F') = R(F)$;*
2. *$\Lambda i = \{0\}$ and $R(F') = R(F)$;*
3. *$\Lambda i = \{0\}$ and $R(F') = R(F) - 1$ (clearly, this case is possible only if $R(F) > 0$).*

Proof. First, notice that any Λ such that $\Lambda i = \{1\}$ leaves F as it was before the transition; thus, trivially, $R(F') = R(F)$. On the other hand, if $\Lambda i = \{0\}$, it can either be that $F' = F$ (and in this case $R(F') = R(F)$ holds trivially) or F' is generated by a failure in F. In the latter case, the failed hyperedge could belong or not to a minimal cut: in the first case, the minimal cut has at least cardinality 1 and $R(F') = R(F) - 1$; otherwise, $R(F') = R(F)$. □

Proof of Proposition 5. Like in the proof of Proposition 3, we have that, for every A with alpha-equivalence class $\mathbf{A}$, it holds that $\mathbf{A} \xrightarrow{\Lambda} \mathbf{G}$ if and only if $\hat{A} \xrightarrow{\Lambda} G$, for some $G \in \mathbf{G}$. So, in the bisimulation game, we shall only consider the transitions of $\hat{A}$.

For the 'if' part, we prove that $\mathcal{S} = \{(\mathbf{F}, \mathbf{G}) \ : \ R(F) = R(G) \wedge F \in \mathbf{F} \wedge G \in \mathbf{G}\}$ is a simulation; indeed, it is easy to see that $R(\hat{A}) = r(A)$. Let $F \xrightarrow{\Lambda} F'$; by Lemma 9, we have only three possibilities for Λ and $R(F')$. In the first two cases, we can always find a Λ' equivalent to Λ such that $G \xrightarrow{\Lambda'} G'$ and $R(G') = R(G)$; then, trivially, $(\mathbf{F}', \mathbf{G}') \in \mathcal{S}$, where $F' \in \mathbf{F}'$ and $G' \in \mathbf{G}'$. Assume instead that $\Lambda i = \{0\}$ and $R(F') = R(F) - 1$; thus, $R(F) > 0$. But then also $R(G) > 0$ and we can find a hyperegde in G whose failure decreases of 1 unit the robustness of G; but such a failure can only have been generated by a Λ' equivalent to Λ and it leads G in a G' whose alpha-equivalence class is related by $\mathcal{S}$ to the alpha-equivalence class of F', by construction of $\mathcal{S}$.

For the 'only if' part, let n be $R(\hat{A})$; thus, there exists a cut of $\hat{A}$ of cardinality n, say $\{v_1, \ldots, v_n\}$. Then, consider the transition Λ_i that makes (the hyperedge associated to) v_i fail; thus, $\hat{A} \xrightarrow{\Lambda_1} F_1 \ldots \xrightarrow{\Lambda_n} F_n$, where $R(F_j) = n - j$. Since $\hat{A} \in \mathbf{A} \sim \mathbf{B} \ni \hat{B}$, it must be that

$\hat{B} \xrightarrow{\Lambda'_1} G_1 \ldots \xrightarrow{\Lambda'_n} G_n$, where Λ'_j is equivalent to Λ_j and the alpha-equivalence class of F_j is bisimilar to the alpha-equivalence class of G_j, for every j. Hence, $\Lambda'_j(i) = \Lambda_j(i) = 0$; so, by Lemma 9, $r(A) = R(\hat{A}) \geq R(\hat{B}) = r(B)$. We can now repeat the same steps starting from $\hat{B}$; we then obtain that $r(B) \geq r(A)$ and easily conclude. $\square$