

# A Logic for Application Level QoS

Dan Hirsch and Alberto Lluch-Lafuente and Emilio Tuosto

*Dipartimento di Informatica,  
Largo Bruno Pontecorvo 3, 56127 Pisa – Italy  
email: {dhirsch,lafuente,etuosto}@di.unipi.it*

## 1 Introduction

Recently, *Service Oriented Computing* (SOC) has been proposed as a paradigm to describe computations of applications on wide area distributed systems. Such systems are very complex and constituted by a varied flora of architectures that typically are heterogeneous, geographically distributed and usually exploit communicating infrastructures whose topology frequently varies and components can, at any moment, connect to or detach from the system. Web services and GRID services may be regarded as SOC and they are receiving particular attention both from academia and industry. An ambitious goal is the automatization of the search-bind cycle so that applications can dynamically chose the “best” service available during the computation. Since the programmer does not completely control the services that her application invokes, it is reasonable to allow her to use declarative mechanisms for expressing the “minimal” requirements on the execution environment.

Recently, *awareness of Quality of Service (QoS)* is emerging as a new exigency in both design and implementation of SOC applications. Remarkably, we *do not* refer to QoS aspects related to low-level performance (as typical e.g. in the community of operating or communication systems). On the contrary, we are concerned with those high-level non-functional features that might interest applications and mainly regard the end-users who perceives QoS not only in connection with low-level performance but as application dependent requirements e.g., the price of a given service, or the payment mode, or else the availability of a resource (e.g., a file in a given format). In the rest of the paper, we intend the acronym QoS to denote application-level QoS.

SOC can be naturally modelled by means of graph-based techniques, where edges represent components and nodes model the communication infrastructure. Edges sharing a node correspond to components that may interact. Systems are modelled as graphs and computations correspond to graph-rewriting. Among other proposals, *hypergraphs* and *synchronised hyperedge replacement* (SHR, for short) have been proposed for modelling distributed systems [11,17]

as a natural declarative framework.

A framework based on *Synchronised Hyperedge Replacement* called SHReQ has been introduced in [24]; SHReQ handles abstract high-level QoS aspects expressed as *constraint-semirings* [3] (c-semirings, for short). Interactions among components are ruled by synchronising them on events that are c-semirings values. Intuitively, the programmer declares the behaviour of each edge  $L$  by specifying a set of *productions*. A production for  $L$  imposes *requirements* to the attachment nodes of  $L$  in order to replace it with a new hypergraph. Such requirements are expressed as elements of a c-semiring and are interpreted as the contribution of  $L$  to the synchronisation. Synchronising requirements is the basic coordination mechanism accounting for evolution of systems and corresponds to the product operation of c-semirings.

In this paper we equip SHReQ with a logic which includes mechanisms to consider the three main dimensions of our systems, namely structure of graphs, behaviour (productions) and QoS. Structural aspects are specified using operators inspired by a spatial logic for graphs (GL) [8], while behavioural ones are tackled with a well-known temporal logic, namely  $\mu$ -calculus. The way the QoS is handled follows the approach of already existing graph and temporal logics for reasoning about QoS in graphs [18] and transition systems [1] (we have slightly adapted the semantics to the SHR framework of SHReQ, since the original logics are defined for simple graphs and transition systems). Formulae are not interpreted as boolean values, but over the domain of the c-semiring modeling the QoS. In other words, the value of a formulae is not just *true* or *false*, but a c-semiring value expressing the *level of satisfiability* of a formula. In that manner one can express the concepts like the QoS level of a path between two given nodes instead of simply expressing whether such a path exists or not.

There exist approaches where spatio-temporal logics for systems involving both structural and behavioural aspects are proposed. First, there are some approaches to reason about rewriting system. For instance, [26] and [2] propose graph transition systems as modeling formalism for rewriting systems. The temporal and spatial aspects are treated differently. The logic of [26] combines the temporal logic LTL with second order quantification over nodes and regular expressions to express path properties, while [2] combines a  $\mu$ -calculus with the Monadic Second-Order (MSO) logic for graphs [13]. In contrast, we interpret formulae directly over SHReQ rewriting systems and use a spatial approach for the structural aspects.

Spatial logics are used to reason about the structure of models, such as heaps [27], trees [9], processes calculi [5,7,6] and graphs [8,14]. The common concept in such approaches is that if a notion of model composition exists (like parallel composition in process calculi or in graph grammar) one can reason about decompositions in the corresponding logic. The usual way is

via a composition operator  $|$ , where  $\phi|\psi$  is satisfied by models that can be decomposed in two sub-models, one satisfying  $\phi$  and another one satisfying  $\psi$ . In graphs, composition is closely related with the second-order quantification over set of edges used in MSO. Indeed, the expressive power of the fixpoint-free fragment of GL have been shown to be included in MSO [14]. The full logic, on the other hand, is able to express properties unlikely to be represented in MSO [14]. It is an open question whether GL subsumes MSO.

Spatial logics for process calculi [5,6,7] include mechanism to reason about name restriction and behaviour. Although node restriction is part of our graph model we neglect mechanism to reason about hidden names for the sake of simplicity.

There are also approaches that interpret temporal logics over domains different than the usual boolean one like boolean algebras [12] or probabilities [15]. There is also a vast number of works regarding the analysis and verification of systems where the focus is on probabilities and time. We cite among others the work on CSL [?], a logic that combines these two aspects. Contrary to such works we do not concentrate on specific issues like time or probabilities but rather consider an abstract representation of QoS by means of a suitable algebraic structure. More precisely, our own contribution to this field of *quantitative* logics is that we propose logics to be defined over constraint-semirings, our formalism for QoS.

In sum, the main novelty of our logic is that it includes mechanisms to reason about structural, behavioural and (c-semiring modeled) QoS aspects. tial approach for the structural aspects.

## Structure of the paper

bla, bla, bla...

## 2 Syntax of Graphs

Given a set of labels  $\mathcal{L}$  ranged over by  $L$  and a set of nodes  $\mathcal{N}$ , a *hyperedge*  $L(x_1, \dots, x_n)$  connects nodes  $x_1, \dots, x_n \in \mathcal{N}$ , where  $L$  has *rank*  $n$  (written  $L : n$ ). We say that  $x_1, \dots, x_n$  are the *attachment nodes* of  $L(x_1, \dots, x_n)$ . *Hypergraphs* are built from ranked hyperedges in  $\mathcal{L}$  and nodes in  $\mathcal{N}$ . In the following, we sometimes use vectors and denote the  $i$ -th component of a  $\mathbf{x}$  by  $\mathbf{x}_i$ , moreover,  $\{\mathbf{x}\} \stackrel{\text{def}}{=} \bigcup_{i \in \{1, \dots, |\mathbf{x}|\}} \{\mathbf{x}_i\}$  and  $|\mathbf{x}|$  is the length of  $\mathbf{x}$ .

**Definition 2.1 (Hypergraphs)** *A hypergraph is a term of the following grammar*

$$G ::= nil \mid L(\mathbf{x}) \mid G \mid G,$$

where  $L : |\mathbf{x}|$  is a hyperedge.

Hereafter, we call hypergraphs (hyperedges) simply graphs (edges) and write  $L(\mathbf{x})$  with the implicit assumption that  $L : |\mathbf{x}|$ . Grammar in Definition 2.1 permits generating the empty graph (denoted by  $nil$ ), graphs with a single edge, graphs built by the parallel composition of graphs and graphs where some nodes are hidden. We use  $n(G)$  to denote the set of nodes of  $G$ . Differently from [24], we neglect node restriction for the sake of simplicity and an easier introduction of the logic.

**Example 2.1** Figure 1(a) represents the hyperedge  $L(a, b, c)$  where wires connecting nodes  $a$ ,  $b$  and  $c$  to  $L$  are called tentacles. The arrowed tentacle individuates the first attachment node. Moving clockwise determines the other tentacles. Figure 1(b) depicts graph  $G = L(y, x, z) | M(x, z)$ .

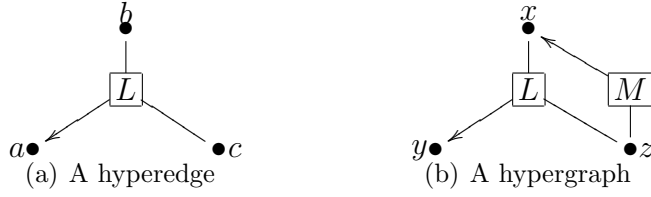


Fig. 1. Hypergraphs

Structural congruence over graph terms avoids cumbersome parenthesis.

**Definition 2.2 (Structural Congruence)** *The structural congruence is the smallest binary relation  $\equiv$  over graph terms that obeys the following axioms:*

$$(G_1 | G_2) | G_3 \equiv G_1 | (G_2 | G_3), \quad G_1 | G_2 \equiv G_2 | G_1, \quad G | nil \equiv G$$

The axioms define associativity, commutativity and identity ( $nil$ ) for operation  $|$ , respectively.

### 2.1 The Ring Case Study

In the following sections we use a running example to exemplify the different topics that are introduced. The example represents a network of rings where gates connect rings (of different sizes) among each other. In other words, the network is configured as a graph where rings represent nodes and gates the arcs among them.

The evolution of the network allows the rings to increase their size while some resources are available. Also, the rings can decide (again based on some resource availability) to create new rings allowing the expansion of the network. The new rings can be created with different sizes. Figure [?] shows (without node names) an instance of the network with three rings, where the one with three components is connected with other two of two components each. Hyperedges labeled with  $l$  identify nodes where gates cannot be created

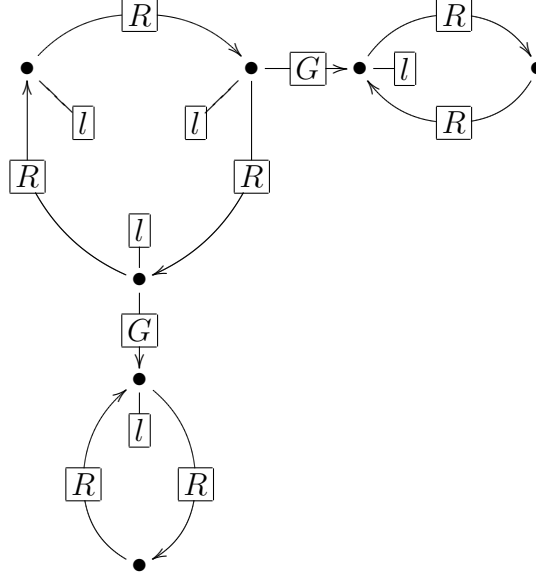


Fig. 2. A network instance

(called *internal nodes*). Also, *larc*-edges are attached (after gate creation) to the nodes where a gate is connected to avoid the creation of a new gate from that nodes. For example, in Figure 5 only the two small rings can create gates to new rings. The nodes with no *l*-edges, can be used to generate new rings and will provide the resource information for their evolution (see Section 3.3).

### 3 Graphs and productions for SHReQ

This section describes SHReQ [24], a calculus based on SHR where c-semiring values are embedded in the rewriting mechanism. In [16], c-semirings have been exploited as a mathematical abstraction for application-level QoS since their algebraic properties can naturally describe QoS values and the usual operations on them. In [25] SHR with mobility of nodes has been generalised by parameterising the rewriting mechanism with respect to a *synchronisation algebra with mobility*. In [24], SHReQ took advantage of the ideas in [16,25] exploiting hypergraphs and SHR with mobility for modelling systems (as in [22,29]) and c-semirings as synchronisation algebras. Hence, the rewriting mechanism of SHReQ is parameterised with respect to a given c-semiring. Basically, values of c-semirings are synchronisation actions so that synchronising corresponds to operate on c-semiring values. Moreover, a hypergraph modelling a system is decorated with c-semiring values on its nodes in order to record quantitative information on the computation of the system. A formal connection can be traced between synchronisation algebras in the sense of [25] and c-semirings, however, this connection is left as future work.

Here, we consider a simplified presentation of SHReQ where mobility is not

considered and, as stated in Section 2, without node restriction for graphs. The first simplification reduces the reconfiguration expressiveness of rewriting systems (i.e., the expressive power is reduced by disallowing node fusion) and the second one simplifies the presentation of the inference system in Table 1 (and the derived proofs). However, these restrictions do not affect the use of c-semiring values as synchronization actions and helps in a clearer introduction of the logic in Section 5, which is our main concern here. For the treatment of node mobility in SHReQ and in SHR, the interested readers are referred to [24] and to [22,29,25].

### 3.1 Weighted graphs

Components (i.e., hyperedges in SHR) impose requirements on their neighbours when participating to a rewriting step. We use c-semirings for expressing those requirements so that rewriting take into account quantitative information on computations. C-semirings have two distinguished features that result very useful in our context. First, the cartesian product of c-semirings is still a c-semiring, hence we can uniformly deal with different types of quantities. Second, the operations of c-semirings provide a partial order on the values and a mechanism of choice. These features make c-semirings suitable for reasoning about multi-criteria QoS issues [16].

**Definition 3.1 (C-semiring [3])** *An algebraic structure  $\langle S, +, \cdot, \mathbf{0}, \mathbf{1} \rangle$  is a constraint semiring if  $S$  is a set ( $\mathbf{0}, \mathbf{1} \in S$ ), and  $+$  and  $\cdot$  are operations on  $S$  satisfying the following properties:*

- $+$  :  $2^S \rightarrow S$  is defined over (possibly infinite) sets of elements of  $S$  as follows<sup>1</sup> :  $\sum \{a\} = a$ ,  $\sum \emptyset = \mathbf{0}$ ,  $\sum S = \mathbf{1}$  and  $\sum (\cup S_i) = \sum \{\sum S_i\}$ , for  $S_i \subseteq S$ ,  $i \geq 0$ .

*The fact that  $+$  is defined over sets of elements, automatically assumes it to be associative, commutative and idempotent. Moreover,  $\mathbf{0}$  is its unit element and  $\mathbf{1}$  is its absorbing element (i.e.,  $a + \mathbf{1} = \mathbf{1}$ , for any  $a \in S$ );*

- $\cdot$  is commutative, associative, distributes over  $+$ ,  $\mathbf{1}$  is its unit element, and  $\mathbf{0}$  is its absorbing element (i.e.,  $a \cdot \mathbf{0} = \mathbf{0}$ , for any  $a \in S$ ). In the rest of the paper we assume that  $\cdot$  is defined over infinite sequences too and use symbol  $\prod$  in postfix notation. Most of the c-semirings used in practice satisfy this.

To enhance readability, operation  $+$  is called *additive operation* and  $\cdot$  is called *multiplicative operation*. The additive operation of a c-semiring induces a partial order on  $S$  defined as  $a \leq_S b \iff a + b = b$ .  $a \leq_S b$  means that  $a$  is more constrained than  $b$  (or that  $b$  "is better than"  $a$ ). Also, it can be shown [3] that  $+$ ,  $\cdot$  are monotone over  $\leq_S$ , the minimal element is  $\mathbf{0}$  and the

<sup>1</sup> When  $+$  is applied to a set with two elements we use  $+$  as binary operator in infix notation, while in all other cases we use symbol  $\sum$  in prefix notation

maximal  $\mathbf{1}$ , and  $\langle S, \leq_S \rangle$  is a complete lattice.

In some instances of a c-semiring, the multiplicative operation is idempotent. This implies, among other things, that  $+$  distributes over  $\cdot$  and  $\langle S, \leq_S \rangle$  is a distributive lattice [3]. In this case, the greatest lower bound (noted as  $\sqcap$ ) corresponds to the multiplicative operation of the c-semiring (the additive operation already corresponds to the least upper bound of a complete lattice).

**Example 3.1** *The following examples introduce some c-semirings together with their application as the formal structure of many QoS attributes.*

- *The boolean c-semiring  $\langle \{true, false\}, \vee, \wedge, false, true \rangle$  can be used to model network and service availability.*
- *The optimization c-semiring  $\langle \mathbb{R}^+, \min, +, +\infty, 0 \rangle$  and the tropical c-semiring  $\langle \mathbb{N}^+, \min, +, +\infty, 0 \rangle$  apply to a wide range of cases, like prices or propagation delay.*
- *Bandwidth can be formalized using the max/min c-semiring  $\langle \mathbb{R}^+, \max, \min, 0, +\infty \rangle$  or also over the naturals  $\langle \mathbb{N}^+, \max, \min, 0, +\infty \rangle$  they can be applied for resource availability.*
- *Performances can be represented by means of the probabilistic c-semiring  $\langle [0, 1], \max, \cdot, 0, 1 \rangle$  or the fuzzy one  $\langle [0, 1], \max, \min, 0, 1 \rangle$ .*
- *The set-based c-semiring  $\langle 2^N, \cup, \cap, \emptyset, N \rangle$  (where  $N$  is a set), can be used for capabilities and access rights.*
- *Security degrees are modeled via the c-semiring  $\langle [0, 1, \dots, n], \max, \min, 0, n \rangle$ , where  $n$  is the maximal security level (unknown) and  $0$  is the minimal one (public).*

**Example 3.2** *As a special case, we can also define c-semirings specifying specific synchronisation mechanisms (for example Hoare, Milner, or Broadcast synchronisation<sup>2</sup>). In this way, we are able to define a general synchronisation policy as a unique c-semiring that combines (using the cartesian product) a classical synchronisation algebra with the QoS aspects of interest. The following c-semiring corresponds to Hoare synchronisation.*

*Given a set of actions  $Act$ , we define  $H = Act \cup \{\mathbf{1}_H, \mathbf{0}_H, \perp\}$ . The Hoare c-semiring on  $H$  is  $\langle H, +_H, \cdot_H, \mathbf{0}_H, \mathbf{1}_H \rangle$  specified as:*

$$\forall a \in Act. a \cdot a = a \tag{1}$$

$$\forall a, b \in Act \cup \{\perp\} : b \neq a \implies a \cdot b = \perp \tag{2}$$

$$\text{plus commutative rules and the ones for } \mathbf{0} \text{ and } \mathbf{1}. \tag{3}$$

<sup>2</sup> See [24] for an example using a Broadcast c-semiring.

The operation  $+_H$  is obtained by extending the  $c$ -semiring axioms for the additive operation with  $a +_H a = a$ , for all  $a \in H$  and  $a +_H b = \perp$ , for all  $a, b \in \text{Act} \cup \{\perp\}$  such that  $b \neq a$ .

With Hoare synchronisation, a synchronisation can take place only when all components attached to the corresponding synchronisation point agree on their actions (i.e., each of these components impose an action and they are all the same). This is reflected in the Hoare  $c$ -semiring product equations.

One restriction needed for using the Hoare  $c$ -semiring for modeling Hoare synchronisation is that the value  $\mathbf{1}_H$  cannot be used as an action in productions. This is due to the fact that for many synchronisation algebras the  $\mathbf{1}$  of a  $c$ -semiring represents the inaction of the algebra, modeling the fact that a component does not participate in a synchronisation. But for Hoare this means that if no synchronisation takes place, then all components remain inactive at that point, however the value  $\mathbf{1}$  does not respect this condition (because it is the product unit element). If inaction rules are needed using Hoare, then it is necessary to add to  $\text{Act}$  a special action, usually called  $\epsilon$ , representing this fact. With action  $\epsilon$ , a node will remain inactive only if all components attached to it remain inactive (using action  $\epsilon$  and by equation 1). If value  $\mathbf{1}_H$  is allowed in rules, this corresponds to a Weak Hoare synchronisation where a certain number of components may agree in the synchronisation, but not necessarily all of them.

Hereafter, we assume a fixed  $c$ -semiring  $\langle S, +, \cdot, \mathbf{0}, \mathbf{1} \rangle$ .

**Definition 3.2 (Weighted graphs)** A weighted graph is a pair  $\Gamma \vdash G$  of a graph  $G$  and a weighting function  $\Gamma$  mapping a finite set of nodes to  $S$  such that  $\text{n}(G) \subseteq \text{dom}_\Gamma$ .

A weighted graph is a graph having values in  $S$  associated to its nodes. We write  $x_1 : s_1, \dots, x_n : s_n \vdash G$  for the weighted graph whose weighting function maps  $x_i$  to  $s_i$ , for any  $i \in \{1, \dots, n\}$ , with the implicit assumptions that nodes  $x_i$  are all distinct and  $\text{n}(G) \subseteq \{x_1, \dots, x_n\}$ . If  $x \notin \text{dom}_\Gamma$ , function  $\Gamma, x : s$  is the updating of  $\Gamma$  on  $x$ .

### 3.2 Productions for weighted graphs

The classical SHR approach is a declarative framework where the behaviour of an edge is specified via a set of *productions* describing the graph to be replaced for the edge, *provided that* some requirements are satisfied by the surrounding environment. A production takes the form  $p : L(\mathbf{x}) \xrightarrow{\Lambda} G$  where  $L(\mathbf{x})$  is a hyperedge,  $G$  a hypergraph and  $\Lambda$  specifies the *requirements*. Roughly,  $p$  states that, in a given graph, an edge labelled  $L$  can be replaced by  $G$  provided that the environment satisfies requirements  $\Lambda$ . Productions of SHReQ have a slightly different definition and interpretation.

**Definition 3.3 (Productions)** *Given a tuple of pairwise distinguished nodes  $\mathbf{x}$  and a graph  $G$  such that  $\{\mathbf{x}\} \subseteq n(G)$ ,  $\chi \triangleright L(\mathbf{x}) \xrightarrow{\Lambda} G$  is a production iff*

- $L$  is an edge label of arity  $|\mathbf{x}|$ ;
- $\chi : \{\mathbf{x}\} \rightarrow S$  is the applicability function;
- $\Lambda : \{\mathbf{x}\} \rightarrow S$  is the communication function.

A SHReQ production, or simply production,  $\chi \triangleright L(\mathbf{x}) \xrightarrow{\Lambda} G$  states that, in order to replace  $L$  with  $G$  in a graph  $H$ , the graph  $H$  must satisfy the conditions expressed by the applicability function  $\chi$  on the attachment nodes of  $L$ . Once  $\chi$  is satisfied in  $H$ ,  $L$  “contributes” to the rewriting by offering  $\Lambda$  in the synchronisation with the other edges connected to its attachment nodes. As will be more clear later,  $\chi$  expressed the *minimal* requirement that the execution environment must satisfy in order to apply the production. Finally, it is understood that the new nodes appearing in  $G$  can be freely renamed (avoiding name-capturing of the nodes in  $\mathbf{x}$ ); moreover, nodes  $\mathbf{x}$  are considered local to the production and can be renamed with fresh names through the whole production.

We remark that, in  $\chi \triangleright L(\mathbf{x}) \xrightarrow{\Lambda} G$ ,  $c$ -semiring values play different roles in  $\chi$  and  $\Lambda$ : in the former, they are interpreted as the minimal requirements that the environment must satisfy for applying the production; in  $\Lambda$  they are the “contribute” that  $L$  yields to the synchronisation with the surrounding edges.

### 3.3 $C$ -semirings and productions for the ring case study

The ring case study is modelled by defining productions on the  $c$ -semiring given by the cartesian product of the Hoare  $c$ -semiring  $\langle H, +_H, \cdot_H, \mathbf{0}_H, \mathbf{1}_H \rangle$  and the *max/min*  $c$ -semiring  $\langle \mathbb{N}^+, \max, \min, 0, +\infty \rangle$ , (Example 3.2 and 3.1). Hoare is used to coordinate synchronisation with respect to the corresponding synchronisation algebra and the *max/min*  $c$ -semiring is used to represent resource availability. The product of *max/min* (i.e. *min*) represents the residual availability of resources after they are consumed during network evolution (i.e., the weights on nodes will decrease after some productions are applied). Also, the weights on nodes will determine when a new ring is created. The  $c$ -semiring  $H$  is defined on  $Act = \{a, b, c\}$ . When clear from the context,  $\mathbf{1}$  (resp.  $\mathbf{0}$ ) denotes the one (resp. the zero) of the cartesian product, also  $\mathbf{1} = (\mathbf{1}_H, +\infty)$  and  $\mathbf{0} = (\mathbf{0}_H, 0)$ .

Figure 3 and Figure 4 show the productions for the case study. For each production, the textual and the graphical representation are given; in the latter case, drawings are simplified by not representing the applicability functions (that appears in the textual representation). Actually, most of them are production schemas corresponding to a set of similar productions. For example, **Create Brother** is a production schema where  $\alpha$  ranges over actions

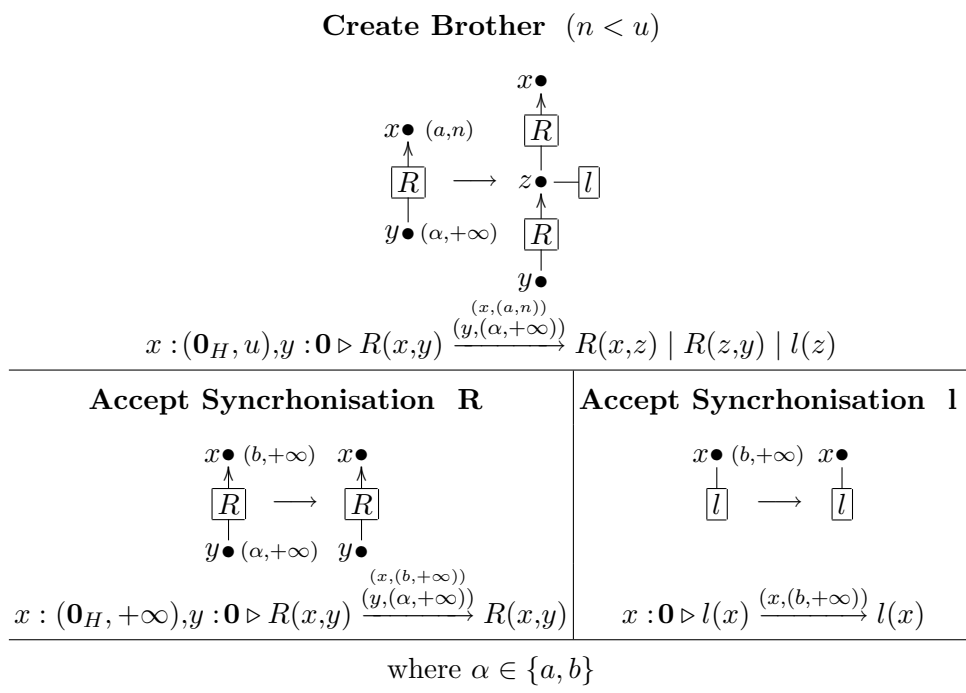


Fig. 3. Productions for ring evolution

$\{a, b\}$  and  $n, u$  must be instantiated for all  $n$  less than  $u$ . Hereafter, instead of writing  $x : s, y : s$  we write  $x, y : s$  whenever possible.

The idea is that the network starts from an initial graph representing a ring with a number of (non- $l$ ) nodes with weights  $(\mathbf{1}_H, u)$  where value  $u$  represents a resource to be consumed. For simplicity, all nodes in newly created rings initially contain the same value  $u$ . The  $l$ -nodes created during ring evolution do not contain useful resource information (noted with value  $+\infty$ , i.e., the one of the c-semiring). Also note that during evolution all internal nodes have always  $+\infty$  in their weight second element.

Productions are grouped in two sets. Figure 3 contains productions modeling the ring evolution and Figure 4 contains productions for the creation and initialization of a gate and a new ring. Detailed comments on the productions follow.

**Create Brother:** This production schema increases the size of the ring with a new component  $R$ . It can be applied over an edge  $R$  that is connected to a non- $l$ -edge on the  $x$  node. Node  $z$  is a new internal node identified by the  $l$ -edge. For applying this production, the applicability condition imposes that the resource weight in the graph for (the node matching)  $x$  has to be greater than or equal to  $u$  (the zeros mean that any weight in the graph satisfies the condition). If the condition is satisfied, the production

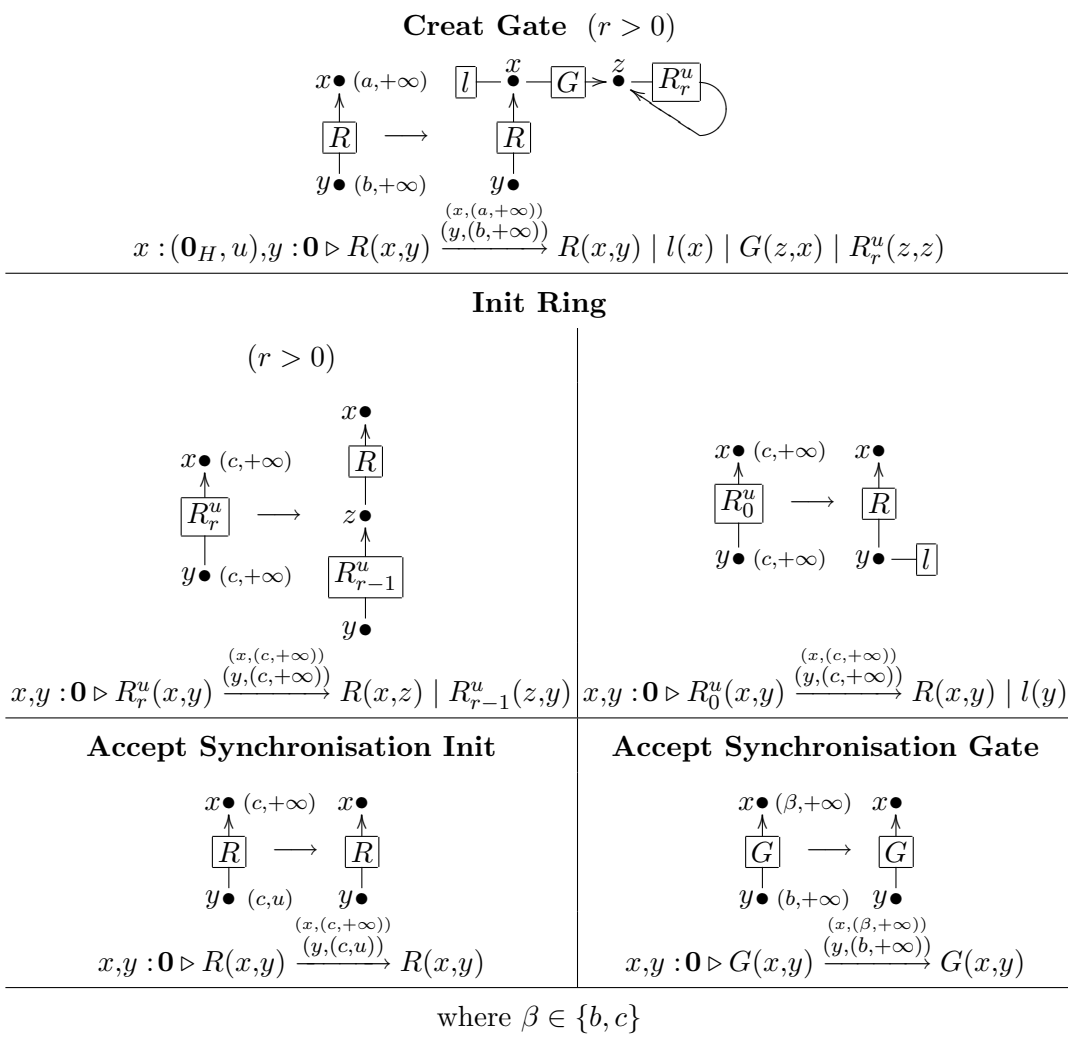


Fig. 4. Productions for creating a new ring

synchronises with its neighbours. On node  $x$  the pair  $(a, n)$  is imposed. A value  $n$  less than  $u$  is selected indicating that there is a cost for applying the production and that the remaining resource value is  $n$ . Action  $a$  assures that the production can only be applied over non-*larc* nodes in the graph matching node  $x$ . This is due to the fact that the only production for edge  $l$  imposes action  $b$  in the connecting node. Then, only productions with action  $b$  will result in a synchronisation on that nodes. The schema also allows to use this production for both cases where node  $y$  is an internal node or not. As will be explained in detail in section 4, note that when  $u = 1$ ,  $n = 0$  and synchronisation cannot take place using this production schema.

**Accept Synchronization  $R$ :** This schema is for ring components connected to two internal nodes ( $\alpha = b$ ) or where node  $y$  is connected to a non- $l$  node ( $\alpha = a$ ). The component synchronises and return to the same state. It is used for agreeing to synchronisation with the rings in charge of creating new components. In this way only one of the rings connected to a node creates a new component (i.e., the one with the arrow point connected to a non- $l$  node). Note that the result of synchronisation on  $y$  for  $\alpha = a$  (using **Create Brother** for the neighbour) will be the product  $(a, n) \cdot (a, +\infty) = (a \cdot_H a, \min(n, +\infty)) = (a, n)$ . This result will be the new weight of the graph node matching  $y$ . We choose to have simple synchronisations to avoid increasing the number of productions complicating the example and its application for the logic.

**Accept Synchronization  $l$ :** This schema is similar to the previous one, and as already explained it is used to assure that ring productions are applied to the right components.

**Create Gate ( $r > 0$ ):** The previous schemas are used for ring evolution. During ring evolution a component connected to a non- $l$  node on its first tentacle and to a  $l$ -node in the second one, can choose to use the remaining resource available (actually, it is a value  $u$  that is at most that value) to create from that node a gate to a new ring. A value  $r$  is selected as the number of non- $l$  nodes for the new ring and  $u$  defines their initial resource value. This information is included in the edge label  $R_r^u$  for the following schemas. An  $l$ -edge is added to convert node  $x$  to an internal one and to assure that no other gate will be created on that node. Note also that synchronisation sets the weight on  $x$  to  $+\infty$  for the second element of the pair. This schema applies to components that have used, at least once, part of their resources for creating a brother. Note that at the end, a gate will always be created on these nodes. The worst case is when  $u = 1$ , where it will be the only applicable production allowing the synchronisation of components and the evolution of the ring. It is worth notice that while a gate is created in this node the rest of the ring can continue its evolution.

**Init Ring:** These are two schemas allowing the initialization of a new ring with respect to the values given by  $r$  and  $u$ . Action  $c$  is used for not allowing the application of rules in Figure 3 (and gate creation) until the ring has finished this phase. When the base case arrives ( $r = 0$ ) it creates the last  $R$  component and a  $l$ -edge connected to the same node as the gate making it internal (this node has weight  $+\infty$  in the pair). At this point we have  $r$  nodes with resource  $u$  and the  $l$ -edge can only synchronise with action  $b$ , therefore, from now on only productions on Figure 3 (plus gate creation) can be used and the new ring can evolve.

**Accept Synchronisation (Init | Gate):** These two schemas are used as the ones in Figure 3 to agree in synchronisation. Also, for **Accept Syn-**

**chronisation Init** it assures that the weight of the new  $r$  nodes contains value  $u$  (as result of action synchronisation among productions).

## 4 Synchronised Rewriting for SHReQ

SHReQ rewriting mechanism relies on c-semirings where addition structure is defined. More precisely, we require that

- there is a set  $Fin$  such that  $Fin \subseteq S$  and  $\mathbf{1} \in Fin$ ;
- there is a set  $NoSync \subseteq S \setminus Fin$  such that  $\forall s \in S : \forall t \in NoSync : s \cdot t \in NoSync$  and  $\mathbf{0} \in NoSync$ .

The intuition is that  $Fin$  contains those values of  $S$  representing events of complete synchronisations. This is a technical expedient from synchronisation algebras with mobility [25] for dealing with restricted nodes. Basically, values in  $Fin$  are those events appearing on restricted nodes that represent a “finished” synchronisation, i.e., an internal synchronisation that does not require any further interaction with the environment. A typical example might be synchronisation actions of process calculi. Set  $NoSync$ , on the contrary, contains the values that represent “impossible” synchronisations. As more clear later, values in  $NoSync$  avoid synchronisations. In the case of the Hoare c-semiring (see example 3.2), all actions in  $Act$  are included in  $Fin_H$  given that all of them can represent a finished synchronization. Set  $NoSync_H$  contains  $\mathbf{0}_H$  and  $\perp$ .

Hereafter, we let  $\Omega$  be a finite multiset over  $\mathcal{N} \times S$ . We write multisets by listing the (occurrences of their) elements in square brackets, e.g.  $[a, a, b]$  is the multiset with two occurrences of  $a$  and one of  $b$  where the order is not important, i.e.,  $[a, a, b] = [a, b, a] = [b, a, a]$ . Multiset membership and difference are expressed by overloading  $\in$  and  $\setminus$ , respectively; the context will always clarify if we are referring to sets or multisets. Multiset union is denoted by  $\uplus$ ; sometimes we also write  $A \uplus B$  to denote the multiset  $[a \mid a \in A] \uplus [b \mid b \in B]$ , where  $A$  or  $B$  is a set.

Before giving SHReQ semantics, we establish some notational conventions:

- $\text{dom}_\Omega = \{x \in \mathcal{N} \mid \exists s \in S : (x, s) \in \Omega\}$ ;
- $\Omega @ x = [(x, s) \mid (x, s) \in \Omega]$ ;
- $\mathcal{W}_\Omega : \text{dom}_\Omega \rightarrow S \quad \mathcal{W}_\Omega : x \mapsto \prod_{(x,s) \in \Omega @ x} s$
- for  $\sigma : \mathcal{N} \rightarrow \mathcal{N}$ ,  $\Omega \sigma = [(\sigma(x), s) \mid (x, s) \in \Omega]$ .

The semantics of SHReQ is a labelled transition system specified with inference rules given on top of *quasi-productions*.

**Definition 4.1 (Quasi-productions)** *The set  $\mathcal{QP}$  of quasi-productions on*

$$(REN) \quad \frac{\chi \triangleright L(\mathbf{x}) \xrightarrow{\Omega} G \in \mathcal{QP} \quad \underline{\Omega} \downarrow \quad x \in \text{dom}_\chi \implies \chi(x) \leq \Gamma(x)}{\Gamma \vdash L(\mathbf{x}) \xrightarrow{\Omega} \Gamma_\Omega^I \vdash G}$$

$$(COM) \quad \frac{x \in \text{dom}_{\Gamma_1} \cap \text{dom}_{\Gamma_2} \implies \Gamma_1(x) = \Gamma_2(x) \quad \Gamma_1 \vdash G_1 \xrightarrow{\Lambda_1} \Gamma'_1 \vdash G'_1 \quad \Gamma_2 \vdash G_2 \xrightarrow{\Lambda_2} \Gamma'_2 \vdash G'_2 \quad \underline{\Lambda_1 \uplus \Lambda_2} \downarrow}{\Gamma_1 \cup \Gamma_2 \vdash G_1 \mid G_2 \xrightarrow{\Lambda_1 \uplus \Lambda_2} \Gamma_1 \cup \Gamma_{2\Lambda_1 \uplus \Lambda_2}^I \vdash G'_1 \mid G'_2}$$

Table 1  
Hypergraph rewriting rules.

$\mathcal{P}$  is defined as the smallest set containing  $\mathcal{P}$  such that

$$\chi \triangleright L(\mathbf{x}) \xrightarrow{\Omega} G \in \mathcal{QP}$$

$$\wedge$$

$$y \in \mathcal{N} \setminus \text{new}(G) \implies \chi' \triangleright L(\mathbf{x}\{y/x\}) \xrightarrow{\Omega\{y/x\}} G\{y/x\} \in \mathcal{QP},$$

where  $x \in \{\mathbf{x}\}$  and  $\chi' : \{\mathbf{x}\} \setminus \{x\} \cup \{y\} \rightarrow S$  is defined as

$$\chi'(z) = \begin{cases} \chi(z), & z \in \{\mathbf{x}\} \setminus \{x, y\} \\ \chi(x) + s, & z = y \wedge (y \in \{\mathbf{x}\} \implies s = \chi(y)) \\ \vee (y \notin \{\mathbf{x}\} \implies s = \mathbf{0}). \end{cases}$$

Intuitively, quasi-productions are obtained by substituting nodes in productions and relaxing the condition that attachment nodes of the left-hand-side should be all different. When  $y$  is substituted for  $x$ ,  $\chi'$  assigns to  $y$  either  $\chi(x) + \chi(y)$  or  $\chi(x)$  depending whether  $y \in \{\mathbf{x}\}$ ; nodes  $z$  not involved in the substitution maintain their constraint  $\chi(z)$ .

**Proposition 4.1**

$$\chi \triangleright L(\mathbf{x}) \xrightarrow{\Omega} G \in \mathcal{QP} \implies \text{dom}_\Omega = \{\mathbf{x}\}.$$

**Definition 4.2 (Communication and weighting)** Let  $\underline{\Omega} : \text{dom}_\Omega \rightarrow S$  be the communication function induced by  $\Omega$  defined as

$$\underline{\Omega}(x) = \mathcal{W}_\Omega(x), \quad (x, s) \in \Omega$$

It is said that  $\underline{\Omega}$  is defined ( $\underline{\Omega} \downarrow$ ) if and only if the following condition holds:

$$\forall x \in \text{dom}_\Omega : |\Omega @ x| > 1 \implies \mathcal{W}_\Omega(x) \notin \text{NoSync}$$

Let  $\Gamma$  be a weighting function such that  $\text{dom}_\Omega \subseteq \text{dom}_\Gamma$  and  $I \subseteq \mathcal{N} \setminus \text{dom}_\Gamma$ , the weighting function induced by  $\Gamma$  and  $\Omega$  is  $\Gamma_\Omega^I : \text{dom}_\Gamma \cup I \rightarrow S$ , defined as

$$\Gamma_\Omega^I : x \mapsto \begin{cases} \mathbf{1}, & x \in I \\ \Gamma(x), & |\Omega @ x| = 1 \\ \mathcal{W}_\Omega(x), & \text{otherwise} \end{cases}$$

For each  $x \in \text{dom}_\Omega$ ,  $\Omega$  computes the requirements resulting from the synchronisation of requirements in  $\Omega @ x$ . More precisely, it multiplies the values according to the c-semiring product. Condition (4.2) avoids synchronisations (and hence rewritings) when a value in *NoSync* is the result of the composition. The weighting function computes the new weights of graphs after the synchronisations induced by  $\Omega$ . New nodes (identified by set  $I$ ) are assigned with  $\mathbf{1}$ , nodes  $x$  upon which no synchronisation took place (i.e.,  $|\Omega @ x| = 1$ ) maintain the old weight while those where synchronisations happen (i.e.,  $|\Omega @ x| > 1$ ) are weighted according to the induced communication function.

We can now define the LTS of weighted graphs.

**Definition 4.3 (Graph transitions)** A SHR with QoS (SHReQ) rewriting system consists of a pair  $(\mathcal{QP}, \Gamma \vdash G)$ , where  $\mathcal{QP}$  is a set of quasi-productions on  $\mathcal{P}$  and  $\Gamma \vdash G$  is the initial weighted graph. The set of transitions of  $(\mathcal{QP}, \Gamma \vdash G)$  is the smallest set obtained by applying the inference rules in Table 1 where,  $I = \text{n}(G) \setminus \text{dom}_\Gamma$ .

Rule (REN) applies quasi-productions to weighted graphs provided that  $\Omega$  is defined and that the weights on the graphs satisfy conditions  $\chi$ , namely,  $\chi(x) \leq \Gamma(x)$ , for all  $x \in \text{dom}_\chi$ . Notice that the communication function and weights in the conclusions are obtained as in Definition 4.2. Similarly, rule (COM) yields the transition obtained by synchronising the transitions of two subgraphs, provided that the (proofs of the) subgraphs assume the same weights on the common nodes.

#### 4.1 SHReQ for the ring case study

This section shows a derivation using productions in Figure 3 and Figure 4 based on the SHReQ semantics (Table 1). We show the graphical representation of the derivation for a better understanding of the example. We omit the proof using the SHReQ rules in Table 1, as their application is quite standard. The example in Figure ?? presents a derivation starting with a ring of two components and ending with the graph in Figure 5. This simple example shows how SHReQ yields a general framework that can deal with system evolution affected by multiple "dimensions" of quality.

First, we define sets *Fin* and *NoSync* of the cartesian product of the

*max/min* and Hoare c-semirings:

- $Fin = \{\mathbf{1}\} \cup \{(a, n) | a \in Act, n > 0\}$ ;
- $NoSync = \{\mathbf{0}\} \cup \{(a, 0) | a \in Act\} \cup \{(\mathbf{0}_H, n) | n \in \mathbb{N}^+\}$ .

Obviously, *Fin* contains all Hoare actions given that they are the result of any complete synchronisation (all  $n > 0$  are valid values). Set *NoSync* contains all pairs with at least one  $\mathbf{0}$  in their components.

Figure ?? shows several steps in the derivation. Some

Instead of reporting the productions for each rewriting step, edge tentacles are decorated with requirements. For the sake of clarity, we index routers to refer them in a simple way, we avoid empty lists of nodes, we represent requirements  $(\mathbf{1}, \langle \rangle)$  with undecorated tentacles and we report only the relevant weights with respect to the considered synchronisation.

In the first derivation,  $U_1$  and  $U_3$  are requesting ambulance assistance to  $R_1$  by synchronizing (on node  $r$ ) **Checking alarm** production for  $R_1$  and **Sending alarm** for  $U_1$  and  $U_3$ . The result of the synchronisation gives  $(\overline{amb}, 1)$  as the new weight of  $r$  and  $U_1$  as the highest priority user (second graph).

The other components do not affect this rewriting step,  $R_2$  and  $U_2$  apply idle productions and  $S$  applies one of its productions (which in this step produce no effect).

The second derivation produces a reconfiguration where  $U_1$  connects to  $S$  by synchronising production **Receiving ambulance assistance** for  $U_1$ , **Forwarding alarm** for  $R_1$  and **Sending ambulance assistance** for  $S$ . User  $U_3$  returns to the initial state with production **Restarting** (the other components apply idle productions). This is shown in the third graph of Figure ?. Moreover, the synchronisation fuses nodes  $z$  and  $x$ .

We remark that all the applicability functions of these productions are satisfied by node weights in the graphs and that productions only ensure that routers choose the highest priority user. For instance, assume that also  $U_2$  requests assistance. It could be the case that the synchronisation among  $R_1$ ,  $R_2$  and  $S$  chooses  $R_2$  instead of  $R_1$ , namely  $U_2$  (instead of  $U_1$ ) will connect to  $S$ . Of course, this could be resolved with productions ruling synchronisation among routers and the server in the style of those among routers and users, however, we prefer not to complicate the example with cumbersome sophistication.

## 5 A Logic for SHReQ

In this section we describe the specification formalism of SHReQ, which consists of a spatio-temporal logic interpreted over c-semiring values.

**Dan, Emilio, questo andrebbe meglio dopo la Def 3.1?** We denote the set of all name nodes (restricted and free) of a graph  $G$  with  $n(G)$ . **E**

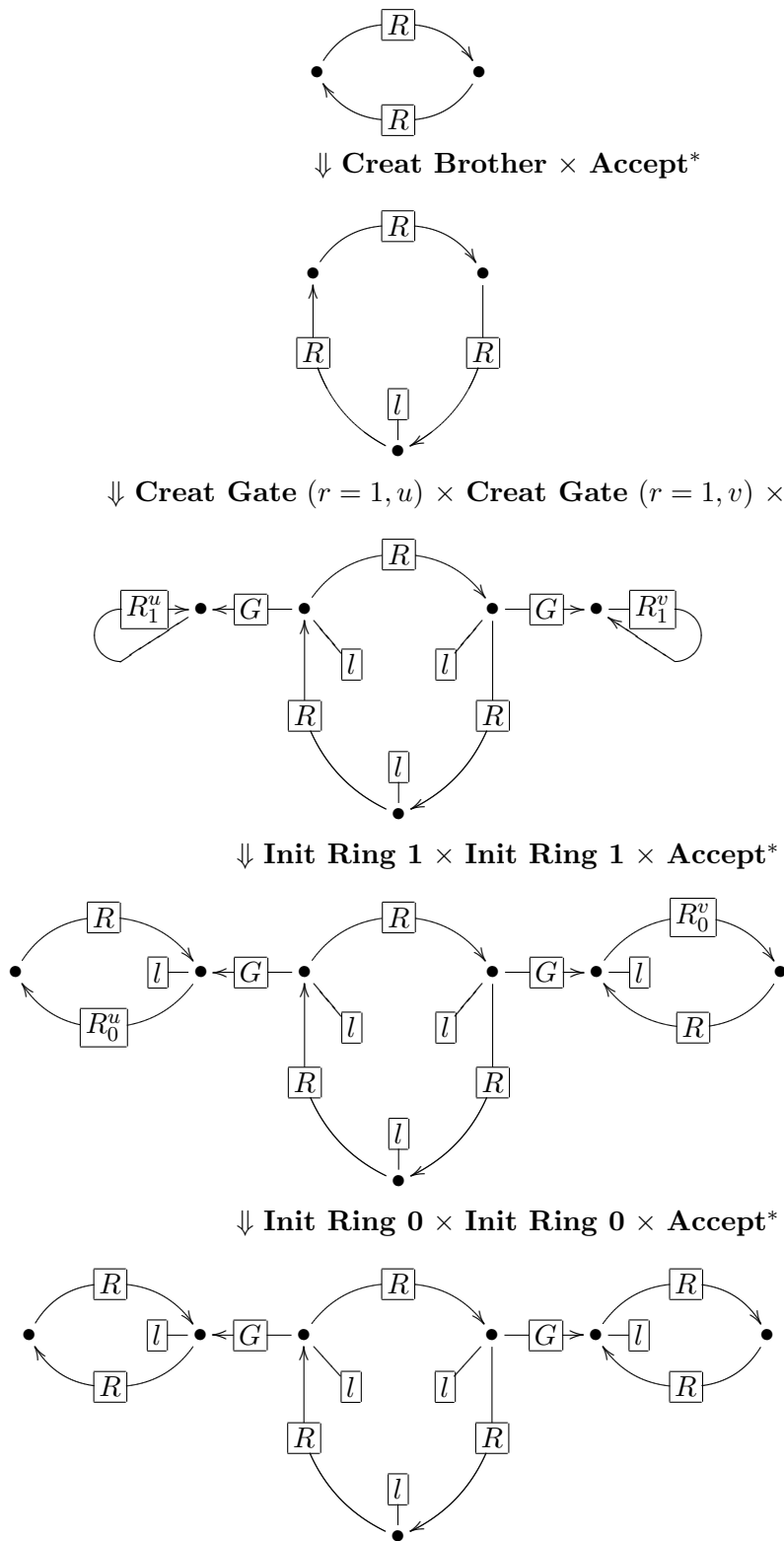


Fig. 5. A network instance

**poi, bisogna spiegare che a ogni nodo viene dato un nome/identit diverso?**

Let  $\mathcal{F}$  be the universe containing all the c-semiring functions  $S^i \rightarrow S$ ,  $i \geq 0$ . Set  $\mathcal{F}$  obviously contains c-semiring addition and multiplication as binary functions, and constant names as zero-adic functions.

A decomposition of a graph  $G$  is an ordered pair of graphs  $G_1, G_2$  such that  $G_1 \mid G_2 \equiv G$ . The set of all decompositions of a graph  $G$  will be denoted by  $\Theta(G)$ . It is not hard to see that the size of  $\Theta(G)$  is exponential in the number of edges of  $G$ .

### 5.1 Logic Syntax.

Let  $V_{\mathcal{N}}$  be a set of node variables and  $V_R$  be a set of recursion variables. Formulae of the graph logic are generated by  $\phi$  in the following grammar:

$$\begin{array}{ll}
\phi ::= nil \mid \Gamma(\xi) \mid L(\bar{\xi}) \mid \phi \mid \phi \mid \phi \parallel \phi & \text{spatial operators} \\
f(\phi, \dots, \phi) & \text{c-semiring operators} \\
\kappa \mathbf{u} . \phi & \text{node quantification} \\
\langle \kappa \rangle \phi & \text{temporal operator} \\
\mathbf{r}(\bar{\xi}) \mid (\mu \mathbf{r}(\bar{\mathbf{u}}) . \phi) \bar{\xi} \mid (\nu \mathbf{r}(\bar{\mathbf{u}}) . \phi) \bar{\xi} & \text{fixpoints}
\end{array}$$

where  $L \in 2^{\mathcal{L}}$ ,  $\mathbf{u} \in V_{\mathcal{N}}$ ,  $\bar{\mathbf{u}} \in V_{\mathcal{N}}^*$ ,  $\xi \in \mathcal{N} \cup V_{\mathcal{N}}$ ,  $\bar{\xi} \in (\mathcal{N} \cup V_{\mathcal{N}})^*$ ,  $f \in \mathcal{F}$ ,  $\kappa \in \{\sum, \prod, \sqcap\}$ , and  $\mathbf{r} \in V_R$ . Obviously, we require  $L : |\Sigma|, |\mathbf{r}| = |\bar{\xi}|$ ,  $|\mathbf{r}| = |\bar{\xi}| = |\bar{\mathbf{u}}|$ . In addition we impose the usual restriction of  $\mathbf{r}(\bar{\xi})$  to occur as operand of a monotone function or under an even number of antimotonic functions in order to guarantee that fixpoints are well defined.

Before giving a formal definition of the semantics of our logic, which is done in the following, we give an intuition of the different syntactic ingredients. With *nil* we characterize graphs with no edges,  $\Gamma(\xi)$  is used to express the weight of nodes, while  $L(\Sigma)$  is used to state whether or not there is an edge with  $\Sigma$  as attachment nodes and with label contained in  $L$ . With  $\phi_1 \mid \phi_2$  we range over all decompositions  $(G_1, G_2)$  of the graph multiplying the evaluation of  $\phi_1$  in  $G_1$  and the evaluation of  $\phi_2$  in  $G_2$ . To all such values, the additive operation is applied. The spatial operator  $\parallel$  is dual with respect to  $+$  and  $\cdot$ . Node quantification evaluates  $\phi$  for each free or hidden<sup>3</sup> node  $\mathbf{u}$  of a graph and then quantifies using  $\kappa$  which is semiring addition, multiplication or glb. C-semiring operations have a straightforward interpretation. The temporal operator  $\langle \kappa \rangle \phi$  applies  $\kappa$  to the values obtained by evaluating  $\phi$  after one

<sup>3</sup> Our logic does not currently distinguish between hidden and free nodes)

transition, and fixpoints with parametrized recursion have the usual meaning (see for [8]), for instance). Observe that ingredients considered as duals are necessary since the absence of a negation operator in c-semirings impedes to derive them from their duals [1]. For instance, a least fixpoint operator cannot be obtained via the greatest fixpoint operator.

## 5.2 Semantics.

The set of all weighted graphs is denoted by  $\mathcal{G}$ <sup>4</sup>. We consider a fixed SHReQ rewriting system  $(\mathcal{QP}, \Gamma \vdash G)$  and interpret a formula as a mapping from  $\mathcal{G}$  into  $S$ , i.e. from the set of all weighted graphs into the domain of the c-semiring. Let  $\sigma : V_{\mathcal{N}} \rightarrow \mathcal{N}$  denote the name environment that maps node variables with nodes<sup>5</sup> and let  $\rho$  be the usual propositional environment mapping recursion variables into functions  $\mathcal{G} \rightarrow S$ . With abuse of notation we let environments be applied to actual names and mappings resulting in the identity, and we abbreviate their application using postfix notation. The interpretation of formulae is as follows:

$$\begin{aligned}
\llbracket nil \rrbracket_{\sigma; \rho}(\Gamma \vdash G) &= G \equiv nil \\
\llbracket \Gamma(\xi) \rrbracket_{\sigma; \rho}(\Gamma \vdash G) &= (\xi\sigma \in n(G)) \cdot \Gamma(\xi\sigma) \\
\llbracket L(\Sigma) \rrbracket_{\sigma; \rho}(\Gamma \vdash G) &= (G \equiv L'(\Sigma\sigma)) \cdot (L' \in L) \\
\llbracket \phi_1 | \phi_2 \rrbracket_{\sigma; \rho}(\Gamma \vdash G) &= \sum_{(G_1, G_2) \in \Theta(G)} \{ \llbracket \phi_1 \rrbracket_{\sigma; \rho}(\Gamma \vdash G_1) \cdot \llbracket \phi_2 \rrbracket_{\sigma; \rho}(\Gamma \vdash G_2) \} \\
\llbracket \phi_1 || \phi_2 \rrbracket_{\sigma; \rho}(\Gamma \vdash G) &= \prod_{(G_1, G_2) \in \Theta(G)} \{ \llbracket \phi_1 \rrbracket_{\sigma; \rho}(\Gamma \vdash G_1) + \llbracket \phi_2 \rrbracket_{\sigma; \rho}(\Gamma \vdash G_2) \} \\
\llbracket f(\phi_1, \dots, \phi_n) \rrbracket_{\sigma; \rho}(\Gamma \vdash G) &= f(\llbracket \phi_1 \rrbracket_{\sigma; \rho}(\Gamma \vdash G), \dots, \llbracket \phi_n \rrbracket_{\sigma; \rho}(\Gamma \vdash G)) \\
\llbracket \kappa \mathbf{u} . \phi \rrbracket_{\sigma; \rho}(\Gamma \vdash G) &= \kappa_{x \in n(G)} \llbracket \phi \rrbracket_{\sigma[x/\mathbf{u}]; \rho}(\Gamma \vdash G) \\
\llbracket \langle \kappa \rangle \phi \rrbracket(\Gamma \vdash G) &= \kappa_{\Gamma \vdash G \Delta \Gamma' \vdash G'} \llbracket \phi \rrbracket(\Gamma' \vdash G') \\
\llbracket \mathbf{r}(\bar{\xi}) \rrbracket_{\sigma; \rho}(\Gamma \vdash G) &= \mathbf{r}\rho(\bar{\xi}\sigma) \\
\llbracket (\mu \mathbf{r}(\bar{\mathbf{u}}) . \phi) \bar{\xi} \rrbracket_{\sigma; \rho}(\Gamma \vdash G) &= \mathit{lfp}(\lambda \mathbf{s} . \lambda \bar{\mathbf{v}} . \llbracket \phi \rrbracket_{\sigma[\bar{\mathbf{v}}/\bar{\mathbf{u}}], \rho[\bar{\mathbf{r}}/\bar{\mathbf{r}}]})(\bar{\xi}\sigma)(\Gamma \vdash G) \\
\llbracket (\nu \mathbf{r}(\bar{\mathbf{u}}) . \phi) \bar{\xi} \rrbracket_{\sigma; \rho}(\Gamma \vdash G) &= \mathit{gfp}(\lambda \mathbf{s} . \lambda \bar{\mathbf{v}} . \llbracket \phi \rrbracket_{\sigma[\bar{\mathbf{v}}/\bar{\mathbf{u}}], \rho[\bar{\mathbf{r}}/\bar{\mathbf{r}}]})(\bar{\xi}\sigma)(\Gamma \vdash G),
\end{aligned}$$

where  $\kappa \in \{\sum, \prod, \prod\}$ . All terms in the right hand side are interpreted over the fixed c-semiring. For example  $\xi\sigma \in X$  returns  $\mathbf{1}$  if  $\xi\sigma$  is an element of  $X$  and  $\mathbf{0}$  otherwise.

One can show that the set of pointwise-ordered total functions of type

<sup>4</sup> Recall that we assume  $\mathcal{N}$ ,  $\mathcal{L}$  and the c-semiring modeling QoS to be fixed.

<sup>5</sup>  $V_{\mathcal{N}}$ ,  $\mathcal{N}$  are required to be disjoint.

$\mathcal{N}^* \rightarrow (\mathcal{G} \rightarrow S)$  are complete lattices or, more generally<sup>6</sup>, c-semirings. It can be also shown that with the mentioned syntactic restrictions functions  $\lambda \bar{y}. \llbracket \phi \rrbracket_{\sigma[\bar{y}/\bar{x}], \rho[\mathbf{s}/\mathbf{r}]}$  are monotone on  $\leq_S$ . Thus, the fixpoints are well defined [28].

## 6 Final Remarks

We presented SHReQ, a formal framework for specifying systems that handle abstract high-level QoS aspects which are becoming more and more important for SOC architectures. SHReQ combines SHR with c-semirings so that the former models mobility and reconfiguration of systems on top of the latter which provide both the mathematics for multi-criteria QoS and the underlying synchronisation policies.

As far as we know, SHReQ is the first to exploit an abstract definition of QoS values as a coordination mechanism. Indeed, in general QoS are either related to low-level aspects of systems or they are simply a notation for describing some non-functional properties of systems. In [22,29], SHR has been extended with mobility for modelling both architectural and programming aspects of mobile distributed applications. SHR with mobility follows a self-organising approach given by the combination of hyperedge rewriting systems for local component specification and constraint solving for coordination. Various facets of SHR have been studied with respect to issues related to distributed systems [22,29,23,16,19]. In [16], c-semirings have been exploited as a mathematical abstraction for application-level QoS since their algebraic properties can naturally describe QoS values and the usual operations on them. For instance, multi-criteria QoS can easily be dealt with cartesian product of c-semirings, which are c-semirings as well. In [16], a middleware expressing quantitative aspects of applications has been proposed and translated into the standard SHR; however, the synchronisation mechanism simply uses point-to-point synchronisation and does not consider quantitative aspects as a primitive coordinating mechanism as presented here. All other SHR frameworks do not consider quantitative aspects.

The work of [21] might be probably considered the closest to our proposal (from a graph transformation standpoint). This approach gives a conceptual model where structural aspects are described in a UML-like meta-model and graph transformation is used for dynamic evolution of systems. Among others, the meta-model provides a *QoS package* that influences the graph transformation rules which are required to respect the QoS package. Despite the similarities, our approach differs from [21] in the fact that in SHReQ the QoS values *are* the rewriting mechanism, not only an additional attribute.

<sup>6</sup> Recall c-semirings form complete lattices [4].

In the area of software architecture, specific QoS aspects (e.g., dependability, performance) have been considered. For instance, in [10,20] run-time monitoring of systems has been considered at the architectural level for handling dynamic self-adaptation that depends on (on-line) performance analysis. These approaches, aside from considering a single QoS aspect instead of application-level multi-criteria and parametric QoS, also apply traditional solutions (e.g., QoS managers) that conceptually differ from SHReQ which distributes the coordination of QoS issues over production specifications.

Modern SOCs usually specify different QoS parameters that depend on applications and should dynamically be integrated and handled. In this context, SHReQ can be generalised to a framework where edges sharing nodes are not supposed to synchronise over the same fixed c-semiring but (defining suitable composition operations among different c-semirings) one could uniformly combine heterogeneous QoS dimensions.

**Dan, Emilio, questo pezzo andrebbe nella conclusione:** We have introduced a logic for reasoning about structural, behavioral and QoS aspects of SHReQ systems. The logic subsumes and generalizes previous approaches for reasoning about graphs [18] and transition systems [1] with QoS. Significant fragments of both logics have been shown to be decidable and model checking algorithms presented (see [1], for instance). A significant fragment, for instance, consists of the special case when the c-semiring is a distributive lattice which occurs when the multiplicative operation is idempotent. This is indeed the case of our scenario. In such cases our full logic can be shown to be decidable. The expressivity and complexity of our logic in general will be subject of future research.

## References

- [1] U. M. Alberto Lluch Lafuente. Quantitative mu-calculus and ctl defined over constraint semirings. *TCS special issue on quantitative aspects of programming languages*, 2005. To appear.
- [2] P. Baldan, A. Corradini, B. König, and B. König. Verifying a behavioural logic for graph transformation systems. In *CoMeta'03*, ENTCS, 2004.
- [3] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *JACM*, 44(2):201–236, March 1997.
- [4] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):201–236, March 1997.
- [5] L. Caires. Behavioral and spatial observations in a logic for the  $\pi$ -calculus. In *FOSSACS'2004*, volume 2987 of *Lecture Notes in Computer Science*, pages 72–87. Springer, 2004.

- [6] L. Caires and L. Cardelli. A spatial logic for concurrency (part II). In *Proceedings of the 13th International Conference on Concurrency Theory*, pages 209–225. Springer-Verlag, 2002.
- [7] L. Caires and L. Cardelli. A spatial logic for concurrency (part I). *Inf. Comput.*, 186(2):194–235, 2003.
- [8] L. Cardelli, P. Gardner, and G. Ghelli. A spatial logic for querying graphs. In *ICALP'2002*, volume 2380 of *Lecture Notes in Computer Science*, pages 597–610. Springer, 2002.
- [9] L. Cardelli, P. Gardner, and G. Ghelli. Manipulating trees with hidden labels. In *Foundations of Software Science and Computation Structures (FOSSACS)*, Lecture Notes in Computer Science, pages 216–232. Springer, 2003.
- [10] M. Castaldi, A. Di Marco, and P. Inverardi. Data driven reconfiguration for performance improvements: a model-based approach. In *Proceedings of RAMSS*, May 2004.
- [11] I. Castellani and U. Montanari. Graph Grammars for Distributed Systems. In H. Ehrig, M. Nagl, and G. Rozenberg, editors, *Proc. 2nd Int. Workshop on Graph-Grammars and Their Application to Computer Science*, volume 153 of *LNCS*, pages 20–38. Springer, 1983.
- [12] M. Chechik, S. Easterbrook, and A. Gurfinkel. Multi-valued symbolic model-checking. *ACM Transactions on Software Engineering and Methodology*, 2003. To appear.
- [13] B. Courcelle. *Handbook of graph grammars and computing by graph transformations*, volume 1 : Foundations, chapter 5: The expression of graph properties and graph transformations in monadic second-order logic, pages 313–400. World Scientific, 1997.
- [14] A. Dawar, P. Gardner, and G. Ghelli. Expressiveness and complexity of graph logic. Technical report, Imperial College, Department of Computing, 2004.
- [15] L. de Alfaro. Quantitative verification and control via the mu-calculus. In *Proceedings of 14th International Conference on Concurrency Theory*, volume 2761 of *Lecture Notes in Computer Science*, pages 102–126. Springer, 2003.
- [16] R. De Nicola, G. Ferrari, U. Montanari, R. Pugliese, and E. Tuosto. A Formal Basis for Reasoning on Programmable QoS. In N. Dershowitz, editor, *International Symposium on Verification – Theory and Practice – Honoring Zohar Manna’s 64th Birthday*, volume 2772 of *LNCS*, pages 436–479. Springer, 2003.
- [17] P. Degano and U. Montanari. A model of distributed systems based on graph rewriting. *JACM*, 34:411–449, 1987.

- [18] G. Ferrari and A. Lluch-Lafuente. A logic for graphs with QoS. In *1st Workshop on Views On Designing Complex Architectures (VODCA)*, Electronic Notes in Theoretical Computer Science. Elsevier, 2004. To appear.
- [19] G. Ferrari, U. Montanari, and E. Tuosto. A LTS Semantics of Ambients via Graph Synchronization with Mobility. In *ICTCS*, volume 2202 of *LNCS*, Torino (Italy), October 4-6, 2001. Springer.
- [20] D. Garlan, B. Schmerl, and J. Chang. Using gauges for architecture-based monitoring and adaptation. In *Working Conference on Complex and Dynamic Systems Architecture*, 2001.
- [21] P. Guo and R. Heckel. Conceptual modeling of styles for mobile systems: A layered approach based on graph transformation. In *Proc. of IFIP TC8 Working Conference on Mobile Information Systems(MOBIS)*, pages 65–78. Springer, 2004.
- [22] D. Hirsch. *Graph Transformation Models for Software Architecture Styles*. PhD thesis, Departamento de Computación, UBA, 2003. <http://www.di.unipi.it/~dhirsch>.
- [23] D. Hirsch, P. Inverardi, and U. Montanari. Reconfiguration of Software Architecture Styles with Name Mobility. In A. Porto and G.-C. Roman, editors, *Coordination 2000*, volume 1906 of *LNCS*, pages 148–163. Springer, 2000.
- [24] Hirsch, Dan and Tuosto, Emilio. SHReQ: A Framework for Coordinating Application Level QoS. In *3rd IEEE International Conference on Software Engineering and Formal Methods (SEFM05)*, Koblenz, Germany, September 2005. IEEE Computer Society Press. To be published.
- [25] I. Lanese and U. Montanari. Synchronization algebras with mobility for graph transformations. In *Proc. FGUC'04 – Foundations of Global Ubiquitous Computing*, ENTCS, 2004. To appear.
- [26] A. Rensink. Towards model checking graph grammars. In M. Leuschel, S. Gruner, and S. L. Presti, editors, *Proceedings of the 3rd Workshop on Automated Verification of Critical Systems*, Technical Report DSSE-TR-2003-2, pages 150–160. University of Southampton, 2003.
- [27] J. Reynolds. Separation logic: A logic for shared mutable data structures. In *LICS 2002*, pages 55–74, 2002.
- [28] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 1955.
- [29] E. Tuosto. *Non-Functional Aspects of Wide Area Network Programming*. PhD thesis, Dipartimento di Informatica, Università di Pisa, May 2003. TD-8/03.