Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 153 (2006) 135–159
www.elsevier.com/locate/entcs

A Logic for Application Level QoS [1,2]

Dan Hirsch[a] Alberto Lluch-Lafuente[a] Emilio Tuosto[b]

[a] *Dipartimento di Informatica, University of Pisa*

[b] *Computer Science Department, University of Leicester*

Abstract

Service Oriented Computing (SOC) has been proposed as a paradigm to describe computations of applications on wide area distributed systems. Awareness of *Quality of Service* (QoS) is emerging as a new exigency in both design and implementation of SOC applications.
We *do not* refer to QoS aspects related to low-level performance and focus on those high-level non-functional features perceived by end-users 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 this paper we present a logic which includes mechanisms to consider the three main dimensions of systems, namely their structure, behaviour and QoS aspects. The evaluation of a formula is a value of a *constraint-semiring* and not just a boolean value expressing whether or not the formula holds. This permits to express not only topological and temporal properties but also QoS properties of systems.
The logic is interpreted on SHReQ, a formal framework for specifying systems that handles abstract high-level QoS aspects combining *Synchronised Hyperedge Replacement* with constraint-semirings.

Keywords: Service oriented computing, synchronized hyperedge replacement, quality of service, logic, c-semirings

1 Introduction

Service Oriented Computing (SOC) is an emerging paradigm to describe computations of *Service Oriented Applications* (SOAs) running on wide area distributed systems. Such systems are very complex and constituted by a varied flora of architectures that typically are heterogeneous, geographically dis-

[1] Partially supported by the Project EC FET – Global Computing 2, IST-2005-16004 SENSORIA and the EC RTN 2-2001-00346 SEGRAVIS (Syntactic and Semantic Integration of Visual Modelling Techniques).
[2] Email: {dhirsch, lafuente}@di.unipi.it, et52@le.ac.uk

doi:10.1016/j.entcs.2005.10.036

tributed and usually exploit communicating infrastructures whose topology frequently varies and to which components can, at any moment, connect to or detach from. Web services and GRID services may be regarded as SOC and they are receiving particular attention both from academia and industry.

An ambitious goal for SOAs is the automatization of the search-bind cycle so that applications can dynamically choose 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 SOAs. 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 perceive 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. System states are modelled as graphs and computations correspond to graph-rewritings. Among other proposals, *hypergraphs* and *Synchronised Hyperedge Replacement* (SHR, for short) have been proposed for modelling distributed systems [9,15] as a natural declarative framework.

A framework based on SHR called SHReQ has been introduced in [19]; 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-semiring values too. In this way, c-semirings provide both the mathematics for multi-criteria QoS and the underlying synchronisation policies. 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 (graph rewriting) and QoS. Structural aspects are specified using operators inspired by a spatial logic for graphs (GL) [7], while behavioural ones are tackled with a well-known temporal logic, namely μ-calculus [16]. The way the QoS is handled follows the approach of already existing graph and temporal logics for reasoning about QoS in graphs [17] and transition systems [21]. Significant fragments of both logics have been shown to be decidable (for finite systems) and model checking algorithms presented [21]. A significant fragment consists of the case when the c-semiring is a distributive lattice which occurs when the multiplicative operation is idempotent. In such cases our full logic can be shown to be decidable. We have adapted the semantics for 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 modelling 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. The expressivity and complexity of our logic in general will be subject of future research.

Structure of the paper: Sections 2, 3 and 4 describe SHReQ together with a running example. Section 5 introduces the logic for SHReQ and applies it to the running example. Final remarks are in Section 6.

2 Syntax of Graphs

Given a set of labels $\mathcal{L}$ ranged over by L and a set of nodes $\mathcal{N}$, a *hyperedge* $L(x_1, ..., x_n)$ connects nodes $x_1, \ldots, x_n \in \mathcal{N}$, where L has *rank* n (written $L : n$). We say that $x_1, ..., x_n$ are the *attachment nodes* of $L(x_1, ..., x_n)$. *Hypergraphs* are built from ranked hyperedges in $\mathcal{L}$ and nodes in $\mathcal{N}$.

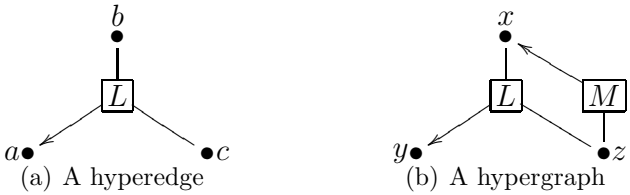
Notation 2.1 *In the following, given a set X ranged over by x, we write $\boldsymbol{x}$ to denote vectors over X. The i-th component of $\boldsymbol{x}$ is denoted by $\boldsymbol{x}_i$ and $\{\!|\boldsymbol{x}|\!\} \stackrel{\text{def}}{=} \bigcup_{i\in\{1,\ldots,|\boldsymbol{x}|\}} \{\boldsymbol{x}_i\}$ and $|\boldsymbol{x}|$ is the length of $\boldsymbol{x}$.*

Definition 2.1 [Hypergraphs] A *hypergraph* is a term of the following grammar

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

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

Hereafter, we call hypergraphs (hyperedges) simply graphs (edges) and write $L(\boldsymbol{x})$ with the implicit assumption that $L : |\boldsymbol{x}|$. The grammar in Definition 2.1 permits generating the empty graph (denoted by *nil*), graphs with a single edge and graphs built by the parallel composition of graphs. We use $\mathrm{n}(G)$ to denote the set of nodes of G. Differently from [19], we neglect node restriction for the sake of simplicity and an easier introduction of the logic.



(a) A hyperedge (b) A hypergraph

Fig. 1. Hypergraphs

Example 2.2 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 $G = L(y, x, z) \mid M(x, z)$.

Structural congruence over graph terms avoids cumbersome parenthesis.

Definition 2.3 [Structural Congruence] The *structural congruence* is the smallest binary relation $\equiv$ over graph terms that obeys the following axioms:

$$(G_1 \mid G_2) \mid G_3 \equiv G_1 \mid (G_2 \mid G_3), \qquad G_1 \mid G_2 \equiv G_2 \mid G_1, \qquad G \mid nil \equiv G$$

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

2.1 The Ring Case Study

In the following sections we exemplify the different topics introduced by means of a running example where "rings" of different sizes are connected by gates yielding a network of rings. A ring consists of a number of nodes connected by some edges forming a cycle and the size of a ring is the number of nodes in it; we write n-ring for a ring of size n.

We assume that a network of rings can become bigger in two ways: by increasing the size of rings or by creating new rings. A ring can augment its size by "consuming" some resources so that the evolution of the network allows rings to increase their size while some resources are available. Also, rings can decide (again depending on resources availability) to create new rings allowing

the expansion of the network. Newly created rings can initially have different sizes.

In Figure 2, rings are represented as nodes connected by hyperedges labelled by R and it shows a network with three rings, where the 4-ring is connected with two rings of size two.

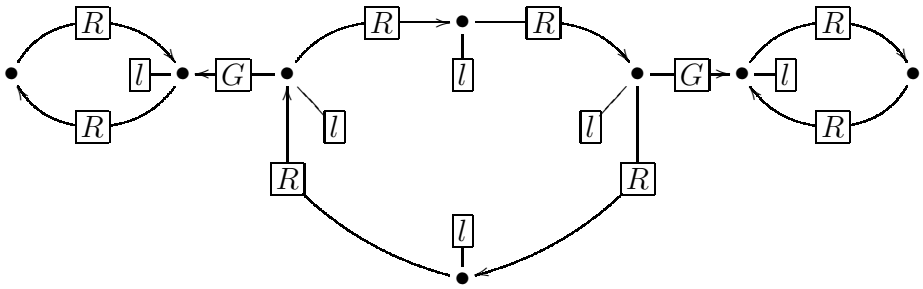


Fig. 2. A network instance

Hyperedges labelled with l (l-edges) prohibit new gates to be attached on the node they insist on; such nodes are called *limited* nodes. As will be made clearer later, on new ring creation, l-edges will be attached to the nodes connected by gate edges (labelled G in Figure 2) avoiding new gates from those nodes. For example, in Figure 2 only the 2-rings can create (gates to) new rings. The nodes with no l-edges, can be used to generate new rings and will be weighted by the amount of resource available for their evolution (see Section 3.3).

3 Graphs and productions for SHReQ

This section describes SHReQ [19], a calculus based on SHR where c-semiring values are embedded in the rewriting mechanism. In [14], 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 [20] SHR with mobility of nodes has been generalised by parameterising the rewriting mechanism with respect to a *synchronisation algebra with mobility*. In [19], SHReQ took advantage of the ideas in [14,20] exploiting hypergraphs and SHR with mobility for modelling systems (as in [18,25]) 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 operating on c-semiring values. Moreover, a hy-

pergraph 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 [20] 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 help 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 [19] and to [18,20,25], respectively.

3.1 Weighted graphs

Components (i.e., hyperedges in SHR) impose requirements on their neighbours when participating into a rewriting step. We use c-semirings to express those requirements so that rewriting takes into account quantitative information on computations. C-semirings have two distinguishing features that are 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 [14].

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:

- $+ : 2^S \to S$ is defined over (possibly infinite) sets of elements of S as follows[3]: $\sum\{a\} = a$, $\sum \emptyset = \mathbf{0}$, $\sum S = \mathbf{1}$ 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

[3] 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.

the paper we assume that $\cdot$ is defined over infinite sequences too and use the symbol $\prod$ in postfix notation. Most of the c-semirings used in practice satisfy this.

To enhance readability, operation $+$ is called the *additive operation* and $\cdot$ is called the *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 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 $\bigsqcap$) corresponds to the multiplicative operation of the c-semiring (the additive operation already corresponds to the least upper bound of a complete lattice).

Example 3.2 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.
- The max/min c-semiring $\langle \mathbb{R}^+, max, min, 0, +\infty \rangle$ can be used to formalize bandwidth, or also the c-semiring over the naturals $\langle \mathbb{N}^+, max, min, 0, +\infty \rangle$ can be applied for resource availability.
- Performance 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 $\langle [0, 1, \ldots, n], max, min, 0, n \rangle$, where n is the maximal security level (unknown) and 0 is the minimal one (public).

Example 3.3 As a special case, we can also define c-semirings specifying specific synchronisation mechanisms (for example Hoare, Milner, or Broadcast synchronisation [4]). In this way, we are able to define a general synchronisation policy as a unique c-semiring that combines (using the cartesian product)

[4] See [19] for an example using a *Broadcast c-semiring*.

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 $H = Act \cup \{\mathbf{1}_H, \mathbf{0}_H, \bot\}$. The *Hoare c-semiring on* H is $\langle H, +_H, \cdot_H, \mathbf{0}_H, \mathbf{1}_H\rangle$ specified as:

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

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

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

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 $a +_H b = \bot$, for all $a, b \in Act \cup \{\bot\}$ 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 multiplicative equations.

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

Definition 3.4 [Weighted graphs] A *weighted graph* is a pair $\Gamma \vdash G$ of a graph G and a *weighting function* Γ mapping a finite set of nodes to S such that $\mathrm{n}(G) \subseteq \mathrm{dom}_\Gamma$.

A weighted graph is a graph having values in S associated to its nodes. We write $x_1 : s_1, \ldots, x_n : s_n \vdash G$ for the weighted graph whose weighting function maps x_i to s_i, for any $i \in \{1, \ldots, n\}$, with the implicit assumption that nodes x_i are all distinct and $\mathrm{n}(G) \subseteq \{x_1, \ldots, x_n\}$. If $x \notin \mathrm{dom}_\Gamma$, function $\Gamma, x : s$ is the extension of Γ 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 $p : L(\boldsymbol{x}) \xrightarrow{\Lambda} G$ where $L(\boldsymbol{x})$ is a hyperedge, G a hypergraph and Λ 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 Λ. Productions of SHReQ have a slightly different definition and interpretation.

Definition 3.5 [Productions] Given a tuple of pairwise distinguished nodes $\boldsymbol{x}$ and a graph G such that $\{\!|\boldsymbol{x}|\!\} \subseteq \mathrm{n}(G)$, $\chi \triangleright L(\boldsymbol{x}) \xrightarrow{\Lambda} G$ is a *production* iff

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

A SHReQ production, or simply production, $\chi \triangleright L(\boldsymbol{x}) \xrightarrow{\Lambda} G$ 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 χ on the attachment nodes of L. Once χ is satisfied in H, L "contributes" to the rewriting by offering Λ in the synchronisation with the other edges connected to its attachment nodes. As will be made clearer later, χ expresses 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 $\boldsymbol{x}$); moreover, nodes $\boldsymbol{x}$ are considered local to the production and can be renamed with fresh names through the whole production.

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

3.3 C-semirings and productions for the ring case study

Productions for the running example of Section 2.1 rely on the c-semiring $\mathcal{R}$ given by the cartesian product of the Hoare c-semiring $\langle H, +_H, \cdot_H, \mathbf{0}_H, \mathbf{1}_H \rangle$, where $H = \{a, b, c, \mathbf{1}_H, \mathbf{0}_H, \bot\}$ and $\langle \mathbb{N}^+, max, min, 0, +\infty \rangle$, the max/min c-semiring (Example 3.3 and 3.2). The former coordinates the network rewritings and the latter represents resource availability, indeed the product of max/min (i.e., min) computes the residual availability of resources. The unit of the multiplication (resp. addition) of $\mathcal{R}$ is $\mathbf{1} = (\mathbf{1}_H, +\infty)$ (resp. $\mathbf{0} = (\mathbf{0}_H, 0)$). Moreover, weights on nodes will determine when a new ring is created.

The idea is that the network starts from an initial graph representing a ring with a number of (non-limited) nodes with weights $(\mathbf{1}_H, u)$ where value u is the maximal amount of available resource. For simplicity, all nodes in newly created rings initially contain the same value u. The limited nodes created during ring evolution are weighted with $(b, +\infty)$ which is constantly maintained.

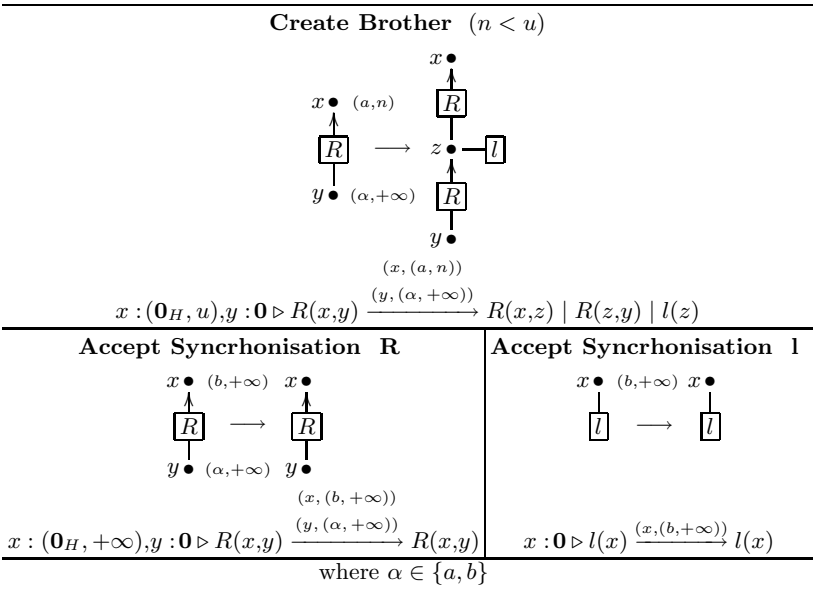


Fig. 3. Productions for ring evolution

Figure 3 and Figure 4 show the productions for the case study giving both textual and graphical representation (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, α ranges over actions $\{a, b\}$ and **Create Brother** is a production schema where $n < u$.

Figure 3 contains productions modelling the ring evolution while Figure 4 contains productions for the creation and initialization of a gate and a new ring. Detailed comments on the productions follow.

Schema **Create Brother** increases the size of the ring by adding a R-edge as represented in the graph on the right-hand-side. The applicability condition on x, $(x : \mathbf{0}_H, u)$, means that the graph weight in matching node of x must be greater than or equal to u (i.e., there is at least a resource value u available). Then, edge $R(x, y)$ imposes on node x an action (a, n) such that u has to be greater than n. Action a assures that the production can only be applied over non limited nodes since the only production for l-edges imposes action b, which cannot synchronise with a. Notice that node z is a new limited node since a l-edge is connected to it. These productions apply regardless of y being a limited node or not.

Schema **Accept Synchronization** R is used to agree to synchronisation with the rings in charge of creating new components. In this way, only one of the R-edges connected to a non limited node creates a new component. The result of synchronising on y for $\alpha = a$ (using **Create Brother** for the

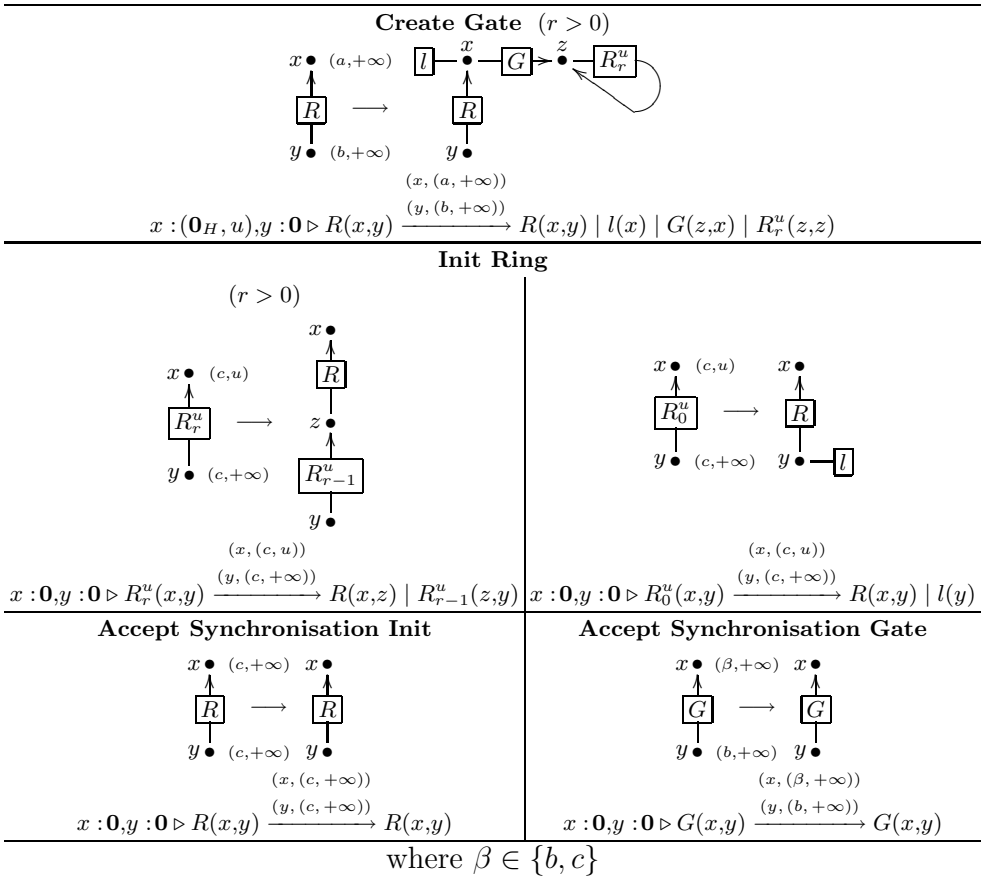


Fig. 4. Productions for creating a new ring

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 and represents the remaining available resource. Note that the condition on x ($x : (\mathbf{0}_H, +\infty)$) assures that x must be a limited node. Finally, **Accept Synchronization** l simply avoids that limited nodes create new brothers and gates.

Create Gate $(r > 0)$ The previous (schema of) productions are used for ring evolution. An edge $R(x,y)$ where x is non limited and y is limited, can consume the remaining resource on x to create from it a gate to a new ring. A value r is non-deterministically chosen as the size of the new ring and u is the initial weight of its nodes (edge R^u_r). Once a new gate is connected to x, it is converted into a limited node by adding an edge $l(x)$. Note that if all resources in a node are consumed using **Create Brother**, we arrive to the base case $u = 1$ and **Create Gate** will be the only applicable production allowing component synchronisation. It is worth noticing that while a gate is created, the rest of the ring can continue its evolution.

Schemas **Init Ring** inductively initialise a new ring to the intended size and to the initial weights of nodes (action (c, u) on node x). Action c avoids applying rules in Figure 3 as well as **Create Gate** until the ring has finished this phase. When the intended size of the ring is obtained ($r = 0$), R_0^u is turned into a R-edge and a l-edge is attached to the same node as the gate. 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 and **Create Gate** can be used. Finally, **Accept Synchronisation Init** and **Accept Synchronisation Gate** are used as for making R and G edges to allow ring initialisation.

4 Synchronised Rewriting for SHReQ

SHReQ rewriting mechanism relies on c-semirings where additional structure is defined. More precisely, we require that there is a set $NoSync \subseteq S$ such that $\forall s \in S : \forall t \in NoSync : s \cdot t \in NoSync$ and $\mathbf{0} \in NoSync$. The intuition is that $NoSync$ is the set of values representing "impossible" synchronisations. In the case of the Hoare c-semiring (Example 3.3), set $NoSync_H$ contains $\mathbf{0}_H$ and $\perp$ while for the cartesian product of the max/min and Hoare c-semirings it is $NoSync = \{\mathbf{0}, \perp\} \cup \{(a, 0) | a \in Act\} \cup \{(\mathbf{0}_H, n) | n \in \mathbb{N}^+\}$.

Before giving SHReQ semantics, we establish some notational conventions. We let Ω 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. Moreover,

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

4.1 SHReQ Semantics

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

$$\chi \triangleright L(\boldsymbol{x}) \xrightarrow{\Omega} G \in \mathcal{QP}$$
$$\wedge$$
$$y \in \mathcal{N} \setminus \mathrm{new}(G) \implies \chi' \triangleright L(\boldsymbol{x}\{^{y}/_{x}\}) \xrightarrow{\Omega\{^{y}/_{x}\}} G\{^{y}/_{x}\} \in \mathcal{QP},$$

where $x \in \{\!|\boldsymbol{x}|\!\}$ and $\chi' : \{\!|\boldsymbol{x}|\!\} \setminus \{x\} \cup \{y\} \to S$ is defined as

$$\chi'(z) = \begin{cases} \chi(z), & z \in \{\!|\boldsymbol{x}|\!\} \setminus \{x, y\} \\ \chi(x) + \chi(y), & z = y \wedge y \in \{\!|\boldsymbol{x}|\!\} \\ \chi(x), & z = y \wedge y \notin \{\!|\boldsymbol{x}|\!\}. \end{cases}$$

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, χ' assigns to y either $\chi(x) + \chi(y)$ or $\chi(x)$ depending whether $y \in \{\!|\boldsymbol{x}|\!\}$; nodes z not involved in the substitution maintain their constraint $\chi(z)$.

Proposition 4.2 $\chi \triangleright L(\boldsymbol{x}) \xrightarrow{\Omega} G \in \mathcal{QP} \implies \mathrm{dom}_\Omega = \{\!|\boldsymbol{x}|\!\}$.

Definition 4.3 [Communication and weighting] Let $\underline{\Omega} : \mathrm{dom}_\Omega \to S$ be the *communication function induced by* Ω defined, for $(x, s) \in \Omega$ as $\underline{\Omega}(x) = \mathcal{W}_\Omega(x)$. We say that $\underline{\Omega}$ *is defined* (written as $\underline{\Omega}\downarrow$) iff

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

Let Γ be a weighting function such that $\mathrm{dom}_\Omega \subseteq \mathrm{dom}_\Gamma$ and $I \subseteq \mathcal{N} \setminus \mathrm{dom}_\Gamma$, the *weighting function induced by* Γ *and* Ω is $\Gamma^I_\Omega : \mathrm{dom}_\Gamma \cup I \to S$, defined as

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

For each $x \in \mathrm{dom}_\Omega$, $\underline{\Omega}$ computes the requirements resulting from the synchronisation of requirements in $\Omega@x$. More precisely, it multiplies the values according to the c-semiring multiplication. Condition (4.3) 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 Ω. New nodes (identified by set I) are assigned with $\mathbf{1}$, nodes x upon which no synchronisation took place (i.e.,

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

$$\text{(COM)}\quad \frac{\begin{array}{c} x \in \mathrm{dom}_{\Gamma_1} \cap \mathrm{dom}_{\Gamma_2} \implies \Gamma_1(x) = \Gamma_2(x) \\ \Gamma_1 \vdash G_1 \xrightarrow{\Lambda_1} \Gamma_1' \vdash G_1' \qquad \Gamma_2 \vdash G_2 \xrightarrow{\Lambda_2} \Gamma_2' \vdash G_2' \qquad \underline{\Lambda_1 \uplus \Lambda_2}\downarrow \end{array}}{\Gamma_1 \cup \Gamma_2 \vdash G_1 \mid G_2 \xrightarrow{\Lambda_1 \uplus \Lambda_2} \Gamma_1 \cup \Gamma_{2\,\Lambda_1 \uplus \Lambda_2}^{\ I} \vdash G_1' \mid G_2'}$$

Table 1
Hypergraph rewriting rules.

$|\Omega@x| = 1$) 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 4.4 [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, $I = \mathrm{n}(G) \setminus \mathrm{dom}_\Gamma$.

Rule (REN) applies quasi-productions to weighted graphs provided that $\underline{\Omega}$ is defined and that the weights on the graphs satisfy conditions χ, namely, $\chi(x) \leq \Gamma(x)$, for all $x \in \mathrm{dom}_\chi$. Notice that the communication function and weights in the conclusions are obtained as in Definition 4.3. 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.

4.2 SHReQ for the ring case study

Now, we show a derivation using productions in Figure 3 and Figure 4 based on the SHReQ semantics (Table 1). This simple example shows how SHReQ can deal with system evolution affected by multiple "dimensions" of quality. In section 5.3 we will use this example to present different properties using the logic for SHReQ.

Instead of a pedantic application of the formal rules, the graphical representation of the derivation is chosen to help intuition (the formal derivation can easily be obtained by following the graphical representation). Figure 5 presents a derivation starting with a 2-ring and ending with the graph in Figure 2. Tentacles are decorated with the synchronisation requirements imposed by productions and nodes are decorated with their weights. For the

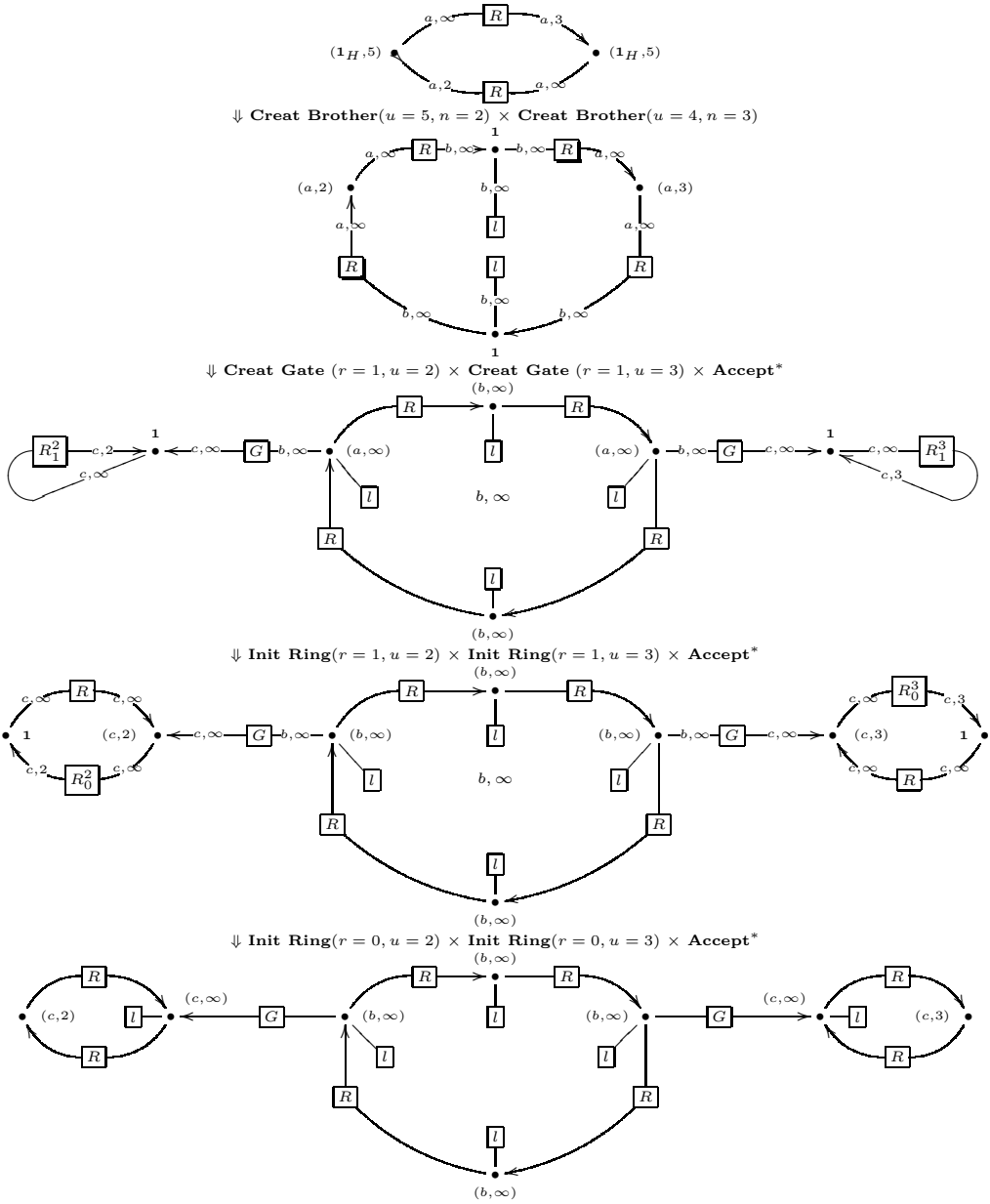


Fig. 5. A derivation

sake of clarity, node names are avoided as node position identifies them during rewriting. Between graphs, the names of the applied productions are indicated together with the instantiation of schema parameters. For the sake of clarity, in the third and fourth graph, the center of the main ring contains a synchronisation action $(b, +\infty)$ indicating that all edge productions in the

main ring impose that action on the nodes.

The derivation starts from a ring of two components with resource value 5. Both components apply production **Create Brother**. The first one chooses $u = 5$ (satisfying condition $5 \leq 5$) and $n = 2$ (meaning that the resource to be consumed is 3). Similarly, the second production chooses $u = 4$ (satisfying condition $4 \leq 5$) and in this case $n = 3$. Synchronised rewriting results in the second graph with four components (new nodes are limited). The resulting synchronisation produces the new weights for the nodes (see second graph) as,

$$(a, 2) = (a, 2) \cdot (a, +\infty) = (a \cdot_H a, \min(2, +\infty))$$
$$(a, 3) = (a, 3) \cdot (a, +\infty) = (a \cdot_H a, \min(3, +\infty)).$$

In the second graph, the two components with the shadow box are the only ones allowed to create brothers or gates. They use all the remaining resources ($u = 2$ and $u = 3$, respectively) to create gates to two 2-rings ($r = 1$); the other edges apply the **Accept** productions. The result of the rewriting step is the third graph, where the new rings are ready to be initialised. Note that the two nodes in the main ring where gates are connected, now are limited.

The last two rewriting steps initialize the new rings with the **Init** productions, obtaining the final graph of Figure 2. Observe how the main ring remains the same applying **Accept** productions, and that, in the last step, productions **Accept Synchronisation Init** ensure that the weights on the non limited nodes of the new rings are $(c, 2)$ and $(c, 3)$, respectively.

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.

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

5.1 Logic Syntax

Let $V_{\mathcal{N}}$ be a set of node variables (disjoint from $\mathcal{N}$ and ranged over by u, v) and V_R be a set of recursion variables (ranged over by $\mathfrak{r}$). Formulae of the graph

logic are generated by ϕ in the following grammar (remind Notation 2.1):

$$\begin{array}{lll}
\phi ::= nil \mid \Gamma(\xi) \mid \mathfrak{L}(\boldsymbol{\xi}) \mid \phi|\phi \mid \phi\|\phi & & \text{spatial operators} \\
\mid f(\phi,\ldots,\phi) & & \text{c-semiring operators} \\
\mid \kappa u.\phi & & \text{node quantification} \\
\mid u = v & & \text{node equality} \\
\mid [\kappa]\,\phi & & \text{temporal operator} \\
\mid \mathfrak{r}(\boldsymbol{\xi}) \mid (\mu\mathfrak{r}(\boldsymbol{u}).\phi)\boldsymbol{\xi} \mid (\nu\mathfrak{r}(\boldsymbol{u}).\phi)\boldsymbol{\xi} & & \text{fixpoints}
\end{array}$$

where $\mathfrak{L} \subseteq \mathcal{L}$ is a finite set of labels, ξ ranges over $\mathcal{N} \cup V_{\mathcal{N}}$, $f \in \mathcal{F}$ and $\kappa \in \{\sum, \prod \bigcap\}$. Obviously, we require $L : |\boldsymbol{\xi}|$ for all $L \in \mathfrak{L}$; moreover, in the last productions for ϕ, $|\boldsymbol{\xi}| = |\boldsymbol{u}|$. In addition, we impose the usual restriction of $\mathfrak{r}(\boldsymbol{\xi})$ to occur as operand of a monotone function or under an even number of antimonotonic 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 Section 5.2, 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 $\mathfrak{L}(\boldsymbol{\xi})$ is used to state whether or not there is an edge with $\boldsymbol{\xi}$ as attachment nodes and with label contained in $\mathfrak{L}$. 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. With $\phi_1|\phi_2$ we range over all decompositions (G_1, G_2) of the graph multiplying the evaluation of ϕ_1 in G_1 and the evaluation of ϕ_2 in G_2. To all such values, the additive operation is applied. As will be made clearer later, the spatial operator $\|$ is dual to $|$. Node quantification evaluates ϕ for each node u of a graph and then quantifies using κ which is semiring addition, multiplication or glb. Node equality and c-semiring operations have a straightforward interpretation. The temporal operator $[\kappa]\,\phi$ applies κ to the values obtained by evaluating ϕ after one transition, and fixpoints with parameterized recursion have the usual meaning (see [7], 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 [21]. 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 $\mathcal{G}$ (recall that we assume $\mathcal{N}$, $\mathcal{L}$ and the c-semiring modeling QoS to be fixed). 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 $\sigma : V_{\mathcal{N}} \to \mathcal{N}$ denote the name environment that maps node variables with nodes and let ρ be the usual propositional environment mapping recursion variables into functions $V_R^* \to (\mathcal{G} \to S)$. With abuse of notation we let environments to 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}
[\![nil]\!]_{\sigma;\rho}(\Gamma \vdash G) &= G \equiv nil \\
[\![\mathfrak{L}(\boldsymbol{\xi})]\!]_{\sigma;\rho}(\Gamma \vdash G) &= \textstyle\sum_{L \in \mathfrak{L}} \{G \equiv L(\boldsymbol{\xi}\sigma)\} \\
[\![\Gamma(\xi)]\!]_{\sigma;\rho}(\Gamma \vdash G) &= (\xi\sigma \in \mathrm{dom}_\Gamma) \cdot \Gamma(\xi\sigma) \\
[\![\xi = \xi']\!]_{\sigma;\rho}(\Gamma \vdash G) &= \xi\sigma = \xi'\sigma \\
[\![f(\phi_1, \ldots, \phi_n)]\!]_{\sigma;\rho}(\Gamma \vdash G) &= f([\![\phi_1]\!]_{\sigma;\rho}(\Gamma \vdash G), \ldots, [\![\phi_n]\!]_{\sigma;\rho}(\Gamma \vdash G)) \\
[\![\phi_1 | \phi_2]\!]_{\sigma;\rho}(\Gamma \vdash G) &= \textstyle\sum_{(G_1,G_2) \in \Theta(G)} \{[\![\phi_1]\!]_{\sigma;\rho}(\Gamma \vdash G_1) \cdot [\![\phi_2]\!]_{\sigma;\rho}(\Gamma \vdash G_2)\} \\
[\![\phi_1 \| \phi_2]\!]_{\sigma;\rho}(\Gamma \vdash G) &= \textstyle\prod_{(G_1,G_2) \in \Theta(G)} \{[\![\phi_1]\!]_{\sigma;\rho}(\Gamma \vdash G_1) + [\![\phi_2]\!]_{\sigma;\rho}(\Gamma \vdash G_2)\} \\
[\![\kappa_u\ \phi]\!]_{\sigma;\rho}(\Gamma \vdash G) &= \kappa_{x \in \mathrm{dom}_\Gamma} [\![\phi]\!]_{\sigma[x/u];\rho}(\Gamma \vdash G) \\
[\![[\kappa]\ \phi]\!]_{\sigma;\rho}(\Gamma \vdash G) &= \kappa_{\Gamma \vdash G \xrightarrow{\Lambda} \Gamma' \vdash G'} [\![\phi]\!](\Gamma' \vdash G') \\
[\![\mathfrak{r}(\boldsymbol{\xi})]\!]_{\sigma;\rho}(\Gamma \vdash G) &= \mathfrak{r}\rho(\boldsymbol{\xi}\sigma) \\
[\![(\mu\mathfrak{r}(\boldsymbol{u}).\phi)\boldsymbol{\xi}]\!]_{\sigma;\rho}(\Gamma \vdash G) &= lfp(\lambda\mathfrak{r}'.\lambda\boldsymbol{v}.[\![\phi]\!]_{\sigma[\boldsymbol{v}/\boldsymbol{u}],\rho[\mathfrak{r} \mapsto \mathfrak{r}']})(\boldsymbol{\xi}\sigma)(\Gamma \vdash G) \\
[\![(\nu\mathfrak{r}(\boldsymbol{u}).\phi)\boldsymbol{\xi}]\!]_{\sigma;\rho}(\Gamma \vdash G) &= gfp(\lambda\mathfrak{r}'.\lambda\boldsymbol{v}.[\![\phi]\!]_{\sigma[\boldsymbol{v}/\boldsymbol{u}],\rho[\mathfrak{r} \mapsto \mathfrak{r}']})(\boldsymbol{\xi}\sigma)(\Gamma \vdash G),
\end{aligned}
$$

where $\kappa \in \{\sum, \prod, \bigsqcap\}$. All terms in the right hand side are interpreted over a fixed c-semiring.

We now concisely describe the semantics of the logic described above. First, observe that some of the ingredients are purely boolean, in the sense that they return either $\mathbf{1}$ or $\mathbf{0}$, which are interpreted as the truth or the falsity. This is indeed the case of nil, $\mathfrak{L}(\boldsymbol{\xi})$, and $\xi = \xi'$. Formula nil characterises graphs with no edges, namely it holds for graphs structurally congruent to nil. Formulae like $\mathfrak{L}(\boldsymbol{\xi})$ are used to reason about the structure of a graph: if a graph G consists of a single edge labelled with some label $L \in \mathfrak{L}$ then the formula evaluates to $\mathbf{1}$ and otherwise to $\mathbf{0}$. Node equality is trivially interpreted as $\mathbf{1}$ if both nodes are equal and $\mathbf{0}$, otherwise.

Remaining formulae might evaluate to any c-semiring value, as far as subformulae of type $\Gamma(\xi)$ are present. Such a formula evaluates to the QoS as-

sociated to a node $\xi\sigma$ but only if the node belongs to the graph where the formula is being evaluated, otherwise $\mathbf{0}$ is returned.

The interpretation of c-semiring functions is straightforward and notice that the interpretation of $\mathbf{1}$ yields $\mathbf{1}$ for any graph.

The standard boolean interpretation of the spatial formula $\phi_1|\phi_2$ (resp. $\phi_1||\phi_2$) is that it evaluates to true for graphs that can be decomposed into two subgraphs one satisfying ϕ_1 and the other ϕ_2 (resp. one subgraph satisfies ϕ_1 or the other satisfies ϕ_2 regardless the graph decomposition). In our logic the semantics is given by substituting boolean conjunction (resp. disjunction) by c-semiring multiplication (resp. addition). Thus, in a graph G, $\phi_1|\phi_2$ is evaluated by considering all possible decompositions (G_1, G_2) of G. For each decomposition, ϕ_1 and ϕ_2 are evaluated in G_1 and G_2, respectively, and the result is multiplied. All the values obtained are summed up. The value of $\phi_1||\phi_2$ is obtained dually with respect to the addition and multiplication operation of the c-semiring, i.e., for each decomposition, ϕ_1 and ϕ_2 are evaluated in G_1 and G_2, and the result is summed up, and finally all the values obtained are multiplied.

As already said, the main originality of c-semiring based logics is that boolean operators and constants are substituted by c-semiring operations and constants. This is also the case of node and next-time quantifiers. Boolean logics typically consider two forms of node and temporal quantification, namely existential and universal. C-semiring based logics substitute existential and universal quantification with semiring addition and multiplication, respectively. However, in addition, c-semiring based logics consider a third way of quantifiying, namely the greatest lower bound (glb) operation of the implicit lattice. The main reason is that in c-semirings where the multiplicative operation is not idempotent the glb and the multiplicative operation do not coincide, but one might be interested in considering both cases. A clear example is the tropical c-semiring, which can be used to model prices. In such systems one could be interested, for instance, in expressing the cheapest (c-semiring addition), the cumulation (c-semiring multiplication) or the most expensive price (c-semiring glb) amongst a set of services. Thus, for instance the value of $[\kappa]\,\phi$ in a graph G is obtained by considering all possible transitions from G to the resulting graphs G', where ϕ is applied. Then, all the values obtained are quantified using κ.

We now describe the semantic of fixpoints formulae. First of all, we consider closed formulae only. Therefore, when evaluating a fixpoint formula like $(\mu\mathfrak{r}(\boldsymbol{u}).\phi)$ or $(\nu\mathfrak{r}(\boldsymbol{u})).\phi$, function $\lambda\mathfrak{r}'.\lambda\boldsymbol{v}.[\![\phi]\!]_{\sigma[^{\boldsymbol{v}}/_{\boldsymbol{u}}],\rho[\mathfrak{r}\mapsto\mathfrak{r}']}$ is well defined [24] since every variable is bound except for its two parameters which are (i) a mapping $\mathfrak{r}'$ of node vectors into functions with $\mathcal{G}$ as domain and S as co-domain, and

(*ii*) a vector $\boldsymbol{v}$ of nodes. Thus, a fixpoint is a function whose domain is the set of node vector and codomain is the set of mappings of $\mathcal{G}$ into S. The application of a fixpoint to a vector of nodes produces a mapping of weighted graphs into c-semiring values. We refer to [17] and [21] for a deeper discussion on these issues.

5.3 Applying the logic

We now illustrate our logic with a set of formulae expressing properties that are interpreted over the example derivation in section 4.2. Note that some of the following properties are purely boolean in the sense that they evaluate either to **0** or to **1**. In such cases we recommend readers to read the formulae in a boolean sense (considering **0** as *False* and **1** as *True*) and to interpret c-semiring addition as disjunction or existential quantification and multiplication as conjunction or universal quantification.

First, we define a formula that expresses that there is a path in a graph between two nodes u, v made of edges labelled by elements in $\mathfrak{L}$.

$$\mathbf{path}(u, v, \mathfrak{L}) \equiv \mu\mathfrak{r}(u, v).(u = v) \; + \; \sum_{w} \mathfrak{L}(u, w)|\mathfrak{r}(w, v).$$

Formula $\mathbf{path}(u, v, \mathfrak{L})$ states that, in a graph, there is a path from u to v either when u is exactly v (i.e., $u = v$) or when the graph can be decomposed in two subgraphs, one containing an arc from u to a node w and the other a path from w to v, i.e. $\sum_{w} \mathfrak{L}(u, w)|\mathfrak{r}(w, v)$.

A pair of nodes u, v belong to a ring whenever there are two disjoint paths from u to v made of R-edges

$$\mathbf{ring}(u, v) \equiv \mathbf{path}(u, v, \{R\}) \mid \mathbf{path}(v, u, \{R\}). \tag{4}$$

Looking at Figure 5, we can see that as the ring formula is defined over a set containing label R, then its application over two nodes belonging to different rings results in **0** because they are connected with a path that must contain a G-edge.

A tree of rings is a tree where nodes are rings connected by G-edges. From formula (4) we can easily derive another one that characterizes trees of rings, by requiring that (5) every node is in a ring, (6) every node is reachable from any other node, and (7) only two nodes in a ring can belong to a cycle:

$$\mathbf{tree} \equiv \prod_u \mathbf{ring}(u,u) \tag{5}$$

$$\cdot \prod_u \prod_v \mathbf{path}(u,v,\{R,G\}) \ + \ \mathbf{path}(v,u,\{R,G\}) \tag{6}$$

$$\cdot \prod_u \prod_v (\mathbf{path}(u,v,\{R,G\}) \cdot \mathbf{path}(v,u,\{R,G\})) \rightarrow \mathbf{ring}(u,v)) \tag{7}$$

where $\rightarrow$ is equivalent to $\leq_S$ interpreted in a boolean way, but we use a different symbol to enhance the readability of the examples. For example, the first, second and last graphs in Figure 5 satisfy the three requirements of the formula. A graph with two gates connecting the same rings falsifies the third requirement and a non connected graph falsifies the second one. This formula shows how to check the configuration consistency of the network.

Previous examples refer only to structural aspects. Now we state a property regarding temporal issues, namely that every non limited node will eventually have a gate attached to it. We use abbreviation $\mathbf{summation}(\phi)$ defined as $\mu\mathfrak{r}.\phi + [\sum]\,\mathfrak{r}$. Note that in a boolean context formula $\mathbf{summation}(\phi)$ can be thought of as $\mathbf{eventually}(\phi)$, with its usual interpretation, i.e., it results in $\mathbf{1}$ if ϕ gives $\mathbf{1}$ at least once. Thus we write:

$$\prod_u \neg(\{l\}(u) \mid \mathbf{1}) \rightarrow \mathbf{summation}\left(\sum_v (\{G\}(v,u) \mid \mathbf{1})\right) \tag{8}$$

where $\neg$ is the c-semiring function that maps $\mathbf{0}$ into $\mathbf{1}$ and every value distinct from $\mathbf{0}$ to $\mathbf{0}$. The interpretation of formula (8) over the initial graph in Figure 5 will range over its two nodes. It is easy to see that the antecedent of the implication is $\mathbf{1}$ (since no l-edge is present), hence the whole property holds only if the consequent is $\mathbf{1}$ as well (because $\rightarrow$ is $\leq_S$ and $\mathbf{1}$ is the maximum of the c-semiring). According to the first two rewriting steps in Figure 5, the initial graph eventually evolves to a graph containing gates connected to that nodes, namely the consequent is evaluated to $\mathbf{1}$, as required.

Up to this point, all presented formulae have been purely boolean. We now introduce a formula concerning QoS aspects and whose interpretation might return values different from $\mathbf{0}$ or $\mathbf{1}$. Assuming to be applied over a graph forming a ring (i.e., a cycle over R-edges), we consider the property obtaining the highest available resource in the ring, i.e., the maximum over the weights of non limited nodes.

$$\left(\prod_v (\{l\}(v) \mid \mathbf{1}) + \sum_u \Gamma(u) \cdot \neg(\{l\}(u) \mid \mathbf{1})\right) \cdot (\mathbf{0}_H, +\infty)$$

In the case of a ring with all limited nodes the result is $+\infty$ (the last part of the formula $(\mathbf{0}_H, +\infty)$ simply selects the second component of the pair). Note that

we quantify over all nodes, but only those not being attached to a l-edge will be relevant, since subformula $\neg(\{l\}(u) \mid \mathbf{1})$ returns $\mathbf{1}$ for non limited nodes, and $\mathbf{0}$ otherwise. Then, for every node the value of the mentioned subformula is multiplied by the QoS of the node, i.e., by $\Gamma(u)$. The first term of the formula $\prod_v(\{l\}(v) \mid \mathbf{1})$ is necessary for the case when all nodes are limited. For example, interpreting this formula on the left-side ring of the last graph of Figure 5 gives $(\mathbf{0}_H, 2)$. For the ring in the right side it is $(\mathbf{0}_H, 3)$, and for the middle ring it is $(\mathbf{0}_H, +\infty)$ given that all its nodes are limited.

Our last formula states a property that regards structure, behaviour and QoS aspects. As defined by production **Create Gate**, a new ring is created with a value u that will be the initial resource weight of non limited nodes of the ring (see Figure 4). We specify a formula yielding the best value (i.e., the maximum) of the initial resource weight of every ring ever generated by the derivations of an intial graph. First, we construct a formula to obtain the resource of a newly generated ring. This can be done by (9) looking over a graph containing a R_0^u-edge, and (10) after the next rewriting step looking at the weight of the first attachment node of the R-edge that has replaced it:

$$\textbf{resource} \equiv \sum_{w,v}(\{R_0^u\}(w,v) \mid \mathbf{1}) \cdot \tag{9}$$

$$\left(\left[\sum\right](\{R\}(w,v) \mid \mathbf{1}) \cdot \Gamma(w)\right). \tag{10}$$

Then we use the abbreviation $(\textbf{summation}(\textbf{resource}) \cdot (\mathbf{0}_H, +\infty))$ which, actually, computes the best value of **resource** in every reachable state [5] . Thus, it expresses the desired property, i.e., the best initial resource value of every ring ever created. The interpretation of $(\textbf{summation}(\textbf{resource}) \cdot (\mathbf{0}_H, +\infty))$ only to the derivation of Figure 5, finds the two new rings over which the maximum is chosen resulting in $(\mathbf{0}_H, 3)$.

5.4 SHReQ logic in context

In the literature, several approaches propose spatio-temporal logics for systems involving both structural and behavioural aspects. First, there are some approaches to reason about rewriting systems. For instance, [22] and [2] propose graph transition systems as modelling formalism for rewriting systems. The temporal and spatial aspects are treated differently. The logic of [22] combines the temporal logic LTL with second order quantification over nodes and regular expressions to express path properties, while [2] combines a μ-calculus with the Monadic Second-Order (MSO) logic for graphs [11]. In contrast, we

[5] We restrict the schemas to obtain a finite set of finite derivations from an initial graph.

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 [23], trees [8], processes calculi [4,5,6] and graphs [7,12]. 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 ϕ and another one satisfying ψ. 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 [12]. The full logic, on the other hand, is able to express properties unlikely to be represented in MSO [12]. It is an open question whether GL subsumes MSO.

Spatial logics for process calculi [4,5,6] include mechanism to reason about name restriction and behaviour. Although node restriction is part of our graph model we neglect a 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 [10] or probabilities [13]. 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 [1], 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.

6 Final Remarks

We have introduced a logic for reasoning about structural, behavioral and QoS aspects of SHReQ systems [19], a formal framework for specifying systems handling abstract high-level QoS aspects. 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. The logic for SHReQ subsumes and generalizes previous approaches for reasoning about graphs [17] and tran-

sition systems [21] with QoS. Significant fragments of both logics have been shown to be decidable (for finite systems) and model checking algorithms presented (see [21], 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 for finite state space SHReQ systems. The intuition is that, as shown in [21], fixpoints are defined over finite lattice, even if the domain of the c-semiring is infinite. Hence, fixpoint iteration algorithms can be applied. On the other hand, if the c-semiring is not a distributive lattice, such algorithms cannot be applied in general. For instance, [21] gives decidability results and algorithms for some fragments of the logic only. However, we believe to obtain positive results for most of the interesting properties one wishes to express in SHReQ systems. This issues will be subject of future research. Also, we plan to develop suitable verification techniques that exploit the expressivity of SHReQ and its logic.

References

[1] C. Baier, B. Havekort, H. Hermanns, and J.-P. Katoen. Model checking continuous-time markov chains by transient analyisis. In *Computer Aided Verification*, volume 1855 of *LNCS*, pages 358–372. Springer Verlag, 2000.

[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] L. Caires. Behavioral and spatial observations in a logic for the π-calculus. In *FOSSACS'2004*, volume 2987 of *Lecture Notes in Computer Science*, pages 72–87. Springer, 2004.

[5] 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.

[6] L. Caires and L. Cardelli. A spatial logic for concurrency (part I). *Inf. Comput.*, 186(2):194–235, 2003.

[7] 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.

[8] 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.

[9] 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.

[10] M. Chechik, S. Easterbrook, and A. Gurfinkel. Multi-valued symbolic model-checking. *ACM Transactions on Software Engineering and Methodology*, 12(4):371–408, 2003.

[11] 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.

[12] A. Dawar, P. Gardner, and G. Ghelli. Expressiveness and complexity of graph logic. Technical report, Imperial College, Department of Computing, 2004.

[13] 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.

[14] 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.

[15] P. Degano and U. Montanari. A model of distributed systems based on graph rewriting. *JACM*, 34:411–449, 1987.

[16] E. Emerson. *Descriptive Complexity and Finite Models*, chapter Model Checking and the Mu-Calculus. American Mathematical Society, 1997.

[17] 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.

[18] D. Hirsch. *Graph Transformation Models for Software Architecture Styles*. PhD thesis, Departamento de Computación, UBA, 2003. http://www.di.unipi.it/~dhirsch.

[19] D. Hirsch and E. Tuosto. SHReQ: A framework for coordinating application level qos. In *3rd IEEE International Conference on Software Engineering and Formal Methods (SEFM05)*, pages 425–434, Koblenz, Germany, September 2005. IEEE Computer Society Press.

[20] 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.

[21] A. Lluch Lafuente and U. Montanari. Quantitative mu-calculus and ctl defined over constraint semirings. *TCS special issue on quantitative aspects of programming languages*, 2005. To appear.

[22] 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.

[23] J. Reynolds. Separation logic: A logic for shared mutable data structures. In *LICS 2002*, pages 55–74, 2002.

[24] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 1955.

[25] E. Tuosto. *Non-Functional Aspects of Wide Area Network Programming*. PhD thesis, Dipartimento di Informatica, Università di Pisa, May 2003. TD-8/03.