Compositional Modeling of Biological Systems in CospanSpan(Graph) (Extended Version)

Alessandro Gianola[1,2], Stefano Kasangian[3], Desiree Manicardi[4], Nicoletta Sabadini[4], and Simone Tini[4]

[1] Free University of Bozen-Bolzano, Italy
gianola@inf.unibz.it
[2] University of California San Diego (UCSD), USA
[3] Università degli Studi di Milano, Italy
[4] Università degli Studi dell'Insubria, Italy

Abstract. The compositional model CospanSpan(Graph) has been shown to model a variety of phenomena from asynchronous circuits to hierarchy, mobility and coordination. Compositionality, i.e. the property of being an algebraic calculus, is here an essential feature: the basic elements of the model are cospans and spans of graphs, that can be interpreted as automata with states and transitions, as well as interfaces and conditions, algebraically composable. There are two classes of operations on these automata - parallel (or product) and sequential (or sum) operations - yielding a categorical algebra of automata. An expression (or even a recursive equation) in this algebra represents a hierarchical, reconfigurable network of interacting components. In this paper we further investigate the expressiveness of CospanSpan(Graph), modeling biological systems: first, we provide a compositional and timed description of the combined, complex system of the Heart and a Dual Chamber Pacemaker. Then, we consider as a case study the well-known gene regulation system in the Lac Operon in Escherichia coli.

Keywords: Automata · Compositionality · Categories · Open networks · Biological Systems

1 Introduction

The CospanSpan(Graph) model, introduced in [13,12], has been shown to model cleanly a variety of phenomena from asynchronous circuits to hierarchy, mobility and coordination [10]. The elements of the model are cospans and spans of graphs which here we shall call simply *Automata with interfaces.* These automata are an extension of the "classical" finite state automata introduced in the seminal work of McCulloch and Pitts, as a *discrete* model for threshold neurons and neural networks. Automata, since then, have become the standard model for the specification and verification of *sequential* discrete dynamical systems. In recent years we have been assisting to a paradigmatic shift from sequential systems (exemplified by Turing Machines) to *networks* of parallel, interacting components.

Hence, various models of automata with product (of states) have been proposed to represent *interactions* (Zielonka, Petri). These models are rather natural, but unfortunately are not compositional, that is they lack a proper algebra. On the contrary, *compositionality*, i.e. the property of providing an algebraic calculus, is an essential feature of CospanSpan(Graph). In this approach, we provide explicitly operations that combine automata with interfaces and their connectors. Hence, given a syntactic expression, its global semantics can be deduced *only* by the semantics of its constituents, and this is precisely what our algebraic approach guarantees. In order to fully achieve compositionality, also with respect to parallelism, we will have to abandon both the classical (inherently sequential and closed) model of finite state automata and the well established idea of input/output communication for a new paradigm, considering as fundamental the notion of *open systems* with communication interfaces. The operations in the algebra CospanSpan(Graph) can be interpreted in a very natural way as operations on automata with states and transitions, as well as interfaces and conditions. An expression (or even a recursive equation) in this algebra represents a hierarchical, reconfigurable *network of interacting components*.

From the beginning, it has been very natural to consider Automata Theory and Biology as very close disciplines, with a long and very fruitful tradition of reciprocal influences, from robotics to biology-inspired models of computation ([3,14,7]). Unfortunately, whereas we could be reasonable satisfied with Turing Machines (and finite state automata) when dealing with discrete, isolated and sequential computation devices, in the literature there is a lack of a general model for *compositional* biology-inspired automata networks that could play an analogous role. This is crucial for performing verifications tasks as well.

In this paper we would like to focus on the importance of an algebraic approach for the *compositional* description of variable topology networks that leads to a natural formalization of biological systems. Specifically, the purpose of this paper is to investigate further the expressivity of the Cospan-Span model, in particular for the modeling of biological discrete real-time systems. In [8], we gave a rather simple but compositional description of the heart. Here, we provide a complete description of a Dual Chamber Pacemaker following [11,1], but, for the first time, in a *compositional* way. Furthermore, we compose the heart model and the pacemaker model, so obtaining a complete specification of the Heart-Pacemaker system. Finally, we consider as an additional case study the well-known gene regulation system in the Lac Operon of the Escherichia coli bacterium and we give a compositional description of it and its functioning.

2 CospanSpan(Graph): an algebraic formalism for automata networks

In this section we recall the algebra **Span**(**Graph**) and its dual counterpart **Cospan**(**Graph**), as introduced by Benabou in [2].

2.1 The Algebra of Spans

Definition 1. *Given a category* **C** *with finite limits, we define a new category* **Span**(**C**) *describing its objects and arrows. Objects of* **Span**(**C**) *are the same objects of* **C***; arrows of* **Span**(**C**) *from A to B (with A, B objects) are spans, that is pairs of arrows $(f : X \to A, g : X \to B)$ of* **C** *with common domain, often written:*

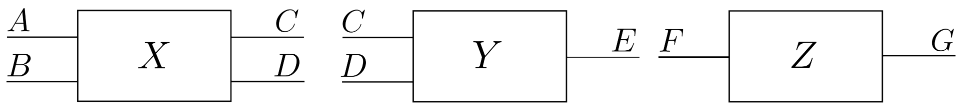
$$\begin{tikzcd}
& X \arrow[dl, "f"'] \arrow[dr, "g"] & \\
A & & B
\end{tikzcd}$$

Composition of spans $(f : X \to A, g : X \to B)$ and $(h : Y \to B, k : Y \to C)$ is by pullback (restricted product):

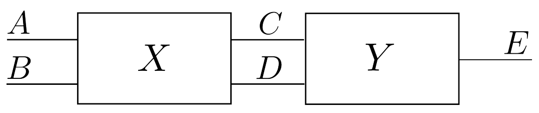
$$\begin{tikzcd}
& & X \times_B Y \arrow[dl] \arrow[dr] & & \\
& X \arrow[dl, "f"'] \arrow[dr, "g"'] & & Y \arrow[dl, "h"] \arrow[dr, "k"] & \\
A & & B & & C
\end{tikzcd}$$

A span $(f : X \to A, g : X \to B)$ will also be written $X : A \to B$, and the composition of span will be indicated with the notation $X \cdot Y : A \longrightarrow C$. The identity span of object A is $(1_A, 1_A)$. The category **Span**(**C**) *is actually symmetric monoidal with the* tensor *product of two spans $(f : X \to A, g : X \to B)$ and $(h : Y \to C, k : Y \to D)$ being $(f \times h : X \times Y \to A \times C, g \times k : X \times Y \to B \times D)$, denoted by $X \times Y$ or $X \otimes Y$. The identity span of object A is $(1_A, 1_A)$.*

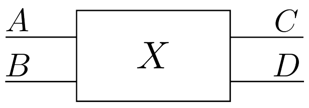
In [13] an informal geometric description (in the style of monoidal category string diagrams) was introduced for the operations in **Span**(**C**). For example, spans $(X \to A \times B, X \to C \times D)$, $(Y \to C \times D, Y \to E)$ and $(Z \to F, X \to G)$ are represented by pictures of *components with ports*:

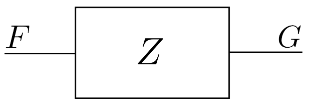


Then the composite of the first two spans is pictured as:



while the tensor of the first and third is pictured as:





In addition, as shown in [10,8], there are constants of the algebra that are pictured as operation on wires and enable the depiction of general circuit diagrams (e.g., a *parallel* feedback [10]). The correspondence between on the one hand constants and operations and on the other hand their geometric representations results in the fact that expressions in the algebra have corresponding circuit or system diagrams.

Span(Graph), a parallel algebra of automata. Surprisingly, **Span**(**C**), when **C** is the category of (finite) directed graphs (**Span**(**Graph**)), provides a very natural mathematical framework for describing the composition of *automata with interfaces* (or communication ports, as in circuit theory). Consider a span of graphs ($\delta_0 : X \to A, \delta_1 : X \to B$). The graph X may be considered as the graph of states and transitions of an automata with interfaces, and it is called the *head* of the span. The graph A is the graph of states and transitions of the combined left ports and B is the graph of states and transitions of the combined right ports. The graph morphism δ_0 associates to a state and to a transition of the automaton X the corresponding state and transition of the left ports A; the morphism δ_1 does the same for the right ports.

For all the examples of this paper the left and right ports have only one state so that we tend to ignore that; then δ_0 and δ_1 are a double labelling of the transitions of the automaton X by transitions on the left ports and transitions on the right ports. More intuitively each transition of the component has an effect on all its interfaces, maybe the null effect ε.

In the case that the left and right ports have one state, the operations of composition and tensor of spans have a simple description in terms of operations on automata. The tensor of two automata has states being pairs of states, one of each automata, and has as transitions pairs of transitions between the corresponding pairs of states. The composition of automata has similarly states being pairs of states, and transitions being pairs of transitions but *only those pairs of transitions whose labels on the connected ports are the same.* In the following, we will call the span composition *parallel composition with communication* and the tensor *parallel composition without communication.*

Remark 1. In [4] *timed actions* with different duration have been considered. Composition is obtained by considering a linear (w.r.t. transitions) number of extra "internal" states. The intended meaning is that a component that interacts with a "faster" one could be still doing an action (hence being in an internal state) when the other one has completed the transition.

2.2 The Algebra of Cospans

Definition 2. *There is a dual construction* $\mathbf{Cospan}(\mathbf{C})$ *for categories* $\mathbf{C}$ *with finite colimits. In fact,* $\mathbf{Cospan}(\mathbf{C}) = \mathbf{Span}(\mathbf{C}^{\mathbf{op}})$*, but it is better seen describing explicitly its objects and arrows. Objects of* $\mathbf{Cospan}(\mathbf{C})$ *are the same as objects of* $\mathbf{C}$*; arrows of* $\mathbf{Cospan}(\mathbf{C})$ *from* A *to* B *are cospans, that is, pairs of arrows* $(f : A \to X, g : B \to X)$ *of* $\mathbf{C}$ *with common codomain, also written as:*

$$\begin{tikzcd}
A \arrow[dr, "f"'] & & B \arrow[dl, "g"] \\
& X &
\end{tikzcd}$$

Composition (which is also called restricted sum) of $(f : A \to X, g : B \to X)$ *and* $(h : B \to Y, k : C \to Y)$ *is by pushout (glued sum), which is the dual operation of the pullback in* $\mathbf{Span}(\mathbf{C})$*. A cospan* $(f : A \to X, g : B \to X)$ *will also be written* $X : A \to B$*, and the composition of cospan will be indicated with the notation* $X + Y : A \longrightarrow C$*. The identity cospan of object* A *is* $(1_A, 1_A)$*.*

Again there are constants of the algebra which are pictured as operation on wires which enable the depiction of joining of wires and sequential feedback [12].

Cospan(Graph), a sequential algebra of automata. Analogously, when $\mathbf{C}$ is the the category of (finite) directed graphs, $\mathbf{Cospan}(\mathbf{Graph})$ provides a sequential calculus for automata that generalizes Kleene's one. Consider a cospan of graphs $(\gamma_0 : E \to X, \gamma_1 : F \to X)$. The graph X may be considered as the graph of states and transitions of an (unlabelled) automaton. The graph E is the graph of *initial* states and transitions and F is the graph of *final* states and transitions. In all the examples considered E and F have only states and not transitions. The graph morphisms γ_0 and γ_1, usually geometrically depicted with dotted lines, are often inclusion morphisms of the initial and final states in X.

As shown in [8,10,17], it is possible to compose the two algebras $\mathbf{Span}(\mathbf{Graph})$ and $\mathbf{Cospan}(\mathbf{Graph})$ in order to get the combined CospanSpan(Graph) algebra, where both sequential and parallel operations are compatible.

3 The heart System in CospanSpan(Graph)

In this section we briefly recall the simplified model of a human heart presented in [8]. For doing so, we took inspiration from [11]. The heart is a muscular organ and it is the engine of the blood flow. Being a physical system, it could be modeled "globally" using continuous time and differential equations, but, we believe, with great difficulty. In our approach we model it *compositionally* as a *discrete* dynamical system. We observe that we focus on the regular behavior of the heart. Considering that the heart system is quite complex, we concentrate first on formalizing its basic components: the heart's interfaces —veins, aorta and pulmonary arteries—, atria, ventricula, tricuspid/mitral valve, sinoatrial node, atrioventricular node and the His bundle.

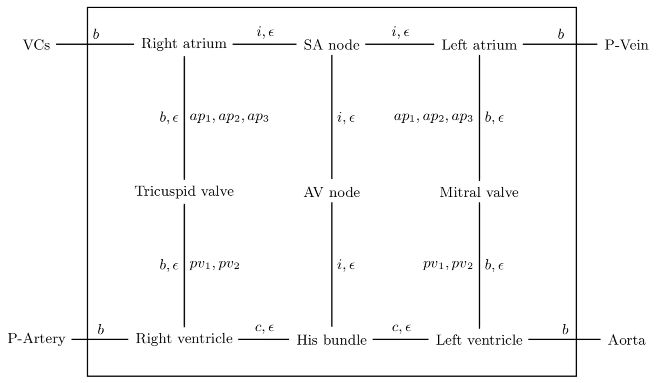
The heart comprehends arterial blood vessels —aorta on the left and pulmonary artery on the right— and two veins. In the right side of the heart, blood continuously enters through the veins and comes out through the pulmonary artery, whereas on the left side enters in the pulmonary vein and comes out from the aorta. So the heart's interfaces are: *(i)* the pulmonary vein and the aorta on the left side of the heart; *(ii)* superior and inferior vena cava and the pulmonary artery on the right side of the heart.

We emphasize that blood is *continuously* going through the heart system, so it would be incorrect to describe its flux in terms of input/output events.

The heart system furthermore consists of four cavity, paired up in two similar subsystems, left and right. Each subsystem includes an atrium, on the top, and a ventricle, on the bottom. Atria related to ventricles through tricuspid —for the right side— and mitral —for the left side— valve. Atria are like a tank of blood coming from veins. After a beat, the blood continues to enter, increasing the internal pressure until a threshold is reached, that causes the tricuspid/mitral valve to open. The blood begins to drain through the tricuspid/mitral valve and at the same time the internal pressure in the atria decreases. There is a further pressure peak in the atria due to the electric signal from the sinoatrial node which causes the atrium's contraction and the blood's flow through the tricuspid/mitral valve.

The sinoatrial node generates the normal rhythmic impulse and distributes this impulse to both the atria, in a simultaneous way. The normal range is 60, 80 beats per minute. The atrioventricular node is primarily responsible for the delay in passing the signal from the atria to the ventricles, whereas the His bundle propagates the impulse to the ventricular heart mass.

For space limits, we only give a high level view of the heart model in the following figure, and its corresponding algebraic expression in CospanSpan(Graph). More details and the description of all the other components can be found in [8].



$$(\text{SVC} \otimes \text{IVC} \otimes \text{SA} \otimes \text{P-Vein}) \cdot (\text{R-Atrium} \otimes \text{AV} \otimes \text{L-Atrium}) \cdot$$

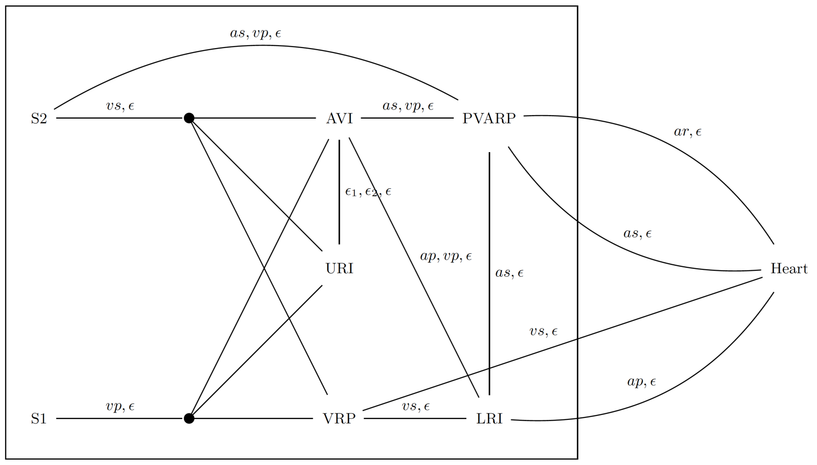
$$\cdot (\text{Tri-Valve} \otimes \text{HIS} \otimes \text{Mitr-Valve}) \cdot (\text{R-Ventr} \otimes \text{L-Ventr}) \cdot (\text{P-Artery} \otimes \text{Aorta})$$

4 Pacemaker in CospanSpan(Graph)

In this section, through the use of timed Cospan-Span(Graph) [4], we provide the model of a pacemaker that communicates with the heart system described in the previous section [8]. A pacemaker is a system that promptly supplies electrical impulses to the heart in order to maintain an appropriate heart rate and also ventricular-atrial synchrony. Different cardiac problems can occur, hence modern pacemakers are used in different ways: each of them has a different labeling. In particular, we model a Dual Chamber Pacemaker DDD − formalized in [11] using UPPAAL − that stimulates both the atrium and the ventricle.

In [11], the Dual Chamber Pacemaker DDD is made up of five components: (i) **LRI** *Lower Rate Interval* (ii) **AVI** *Atrio-Ventricular Interval* (iii) **URI** *Upper Rate Interval* (iv) **PVARP** *Post Ventricular Atrial Refractory Period* and **PVAB** *Post Ventricular Atrial Blanking* (v) **VRP** *Ventricular Refractory Period*. Bear in mind that the functioning of the modeled pacemaker is a first approximation that can be extended, e.g. also considering a different type of pacemaker or using probabilistic Cospan-Span(Graph) formalism [6].

The next picture shows the pacemaker architecture and the communications with our heart system. Unlike [11], we add two components for broadcasting transmission: S1 and S2 - respectively for AP and VS - which transmits the signal to different other components simultaneously.

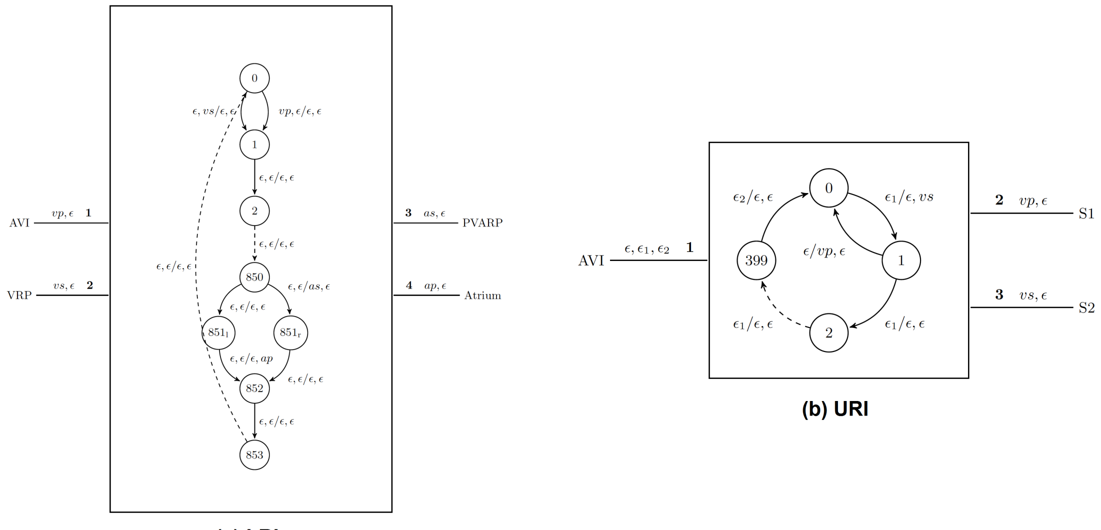


The pacemaker shown here was modeled considering the heart in brachycardia or with a regular beat − with 80 beats per minute. The time used in the timed version of Span(Graph) is discrete with $\Delta = 10$ milliseconds. The constants TAVI, TLRI, TPVARP, TVRP, TURI and TPVAB − described in [11] − which control the duration of the operations, have the following values: (i) **TAVI**: 150 ms; (ii) **TLRI**: 1000 ms; (iii) **TPVARP**: 100 ms; (iv) **TVRP**: 150 ms; (v) **TURI**: 400 ms; (vi) **TPVAB**: 50ms.

In the following, we adopt the convention to denote interfaces (by abuse of notation) with *Component*=\{*labels*\}, where labels are the proper labels of the interfaces and *Component* corresponds to the automaton to be connected. For sake of examples, we describe the **LRI**, **AVI**, **URI** and **VRP** components (the details about the other components can be found here [9]).

Now we describe **LRI** component (Figure 1**(a)**) which has the task of maintaining the heart rate: it models the cycle that defines the longest interval between two ventricular events by resetting the clock when a ventricular event (VS, VP) is received. Furthermore, if it does not detect any atrial event AS, the component delivers the atrial pacing AP after TLRI-TAVI (850 millisecond). As shown in Figure 1**(a)**, the time starts with S0; in addition, LRI tracks the time through the passage from one state to the next. LRI has four interfaces: AVI = $\{vp, \epsilon\}$, VRP = $\{vs, \epsilon\}$, PVARP = $\{as, \epsilon\}$ and Atrium = $\{ap, \epsilon\}$. The transitions are:

$$\begin{array}{cc}
\text{vp},\epsilon/\epsilon,\epsilon : 0 \rightarrow 1 & \epsilon,\text{vs}/\epsilon,\epsilon : 0 \rightarrow 1 \\
\epsilon,\epsilon/\epsilon,\epsilon : 1 \rightarrow 2 & \epsilon,\epsilon/\epsilon,\epsilon : 2 \rightarrow 3 \\
\ldots & \ldots \\
\epsilon,\epsilon/\epsilon,\epsilon : 848 \rightarrow 849 & \epsilon,\epsilon/\epsilon,\epsilon : 849 \rightarrow 850 \\
\epsilon,\epsilon/\text{as},\epsilon : 850 \rightarrow 851_{\text{r}} & \epsilon,\epsilon/\epsilon,\epsilon : 851_{\text{r}} \rightarrow 852 \\
\epsilon,\epsilon/\epsilon,\epsilon : 850 \rightarrow 851_{\text{l}} & \epsilon,\epsilon/\epsilon,\text{ap} : 851_{\text{l}} \rightarrow 852 \\
\epsilon,\epsilon/\epsilon,\epsilon : 852 \rightarrow 853 & \epsilon,\epsilon/\epsilon,\epsilon : 853 \rightarrow 0
\end{array}$$



(a) LRI **(b) URI**

Fig. 1. LRI and URI components

Next, we describe the **AVI** (Figure 2) which maintains the appropriate interval between atrial and ventricular activation so it defines the longest interval between an atrial event and a ventricular event. If AVI does not detect any ventricular event (VS) after an atrial event (AS, AP), within TAVI, then AVI delivers a ventricular stimulation (VP). AVI has five interfaces: LRI = $\{ap, \epsilon\}$, PVARP = $\{as, \epsilon\}$, S2 = $\{vs, \epsilon\}$, URI = $\{\epsilon, \epsilon_1, \epsilon_2\}$ and S1 = $\{vp, \epsilon\}$. The transitions are:

$$\begin{array}{rr}
ap, \epsilon, \epsilon/\epsilon, \epsilon : \text{-1} \to 0 & \epsilon, as, \epsilon/\epsilon, \epsilon : \text{-1} \to 0 \\
\epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon : 0 \to 1 & \epsilon, \epsilon, \epsilon/\epsilon_2, \epsilon : 0 \to 1 \\
\epsilon, \epsilon, \epsilon/\epsilon_2, \epsilon : 1 \to 2 & \epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon : 1 \to 2 \\
\ldots & \ldots \\
\epsilon, \epsilon, \epsilon/\epsilon_2, \epsilon : 149 \to 150 & \epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon : 149 \to 150 \\
\epsilon, \epsilon, vs/\epsilon, \epsilon : 150 \to \text{-1} & \epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon : 150 \to 151 \\
\epsilon, \epsilon, vs/\epsilon, \epsilon : 151 \to \text{-1} & \epsilon, \epsilon, \epsilon/\epsilon_2, \epsilon : 151 \to \text{-1} \\
\epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon : 151 \to 151 & \epsilon, \epsilon, \epsilon/\epsilon, \epsilon : \text{-1} \to \text{-1}
\end{array}$$

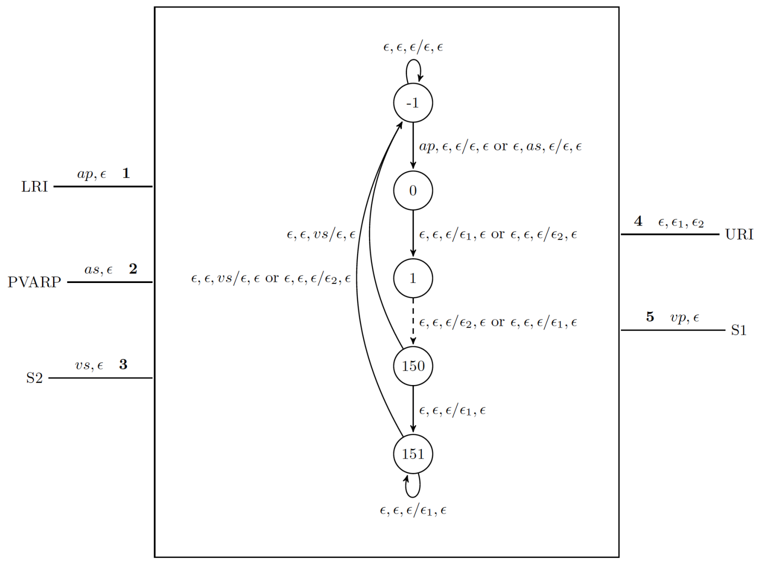

$\epsilon, \epsilon, \epsilon/\epsilon, \epsilon$
-1
$ap, \epsilon, \epsilon/\epsilon, \epsilon$ or $\epsilon, as, \epsilon/\epsilon, \epsilon$
0
$\epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon$ or $\epsilon, \epsilon, \epsilon/\epsilon_2, \epsilon$
1
$\epsilon, \epsilon, \epsilon/\epsilon_2, \epsilon$ or $\epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon$
150
$\epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon$
151
$\epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon$
$\epsilon, \epsilon, vs/\epsilon, \epsilon$
$\epsilon, \epsilon, vs/\epsilon, \epsilon$ or $\epsilon, \epsilon, \epsilon/\epsilon_2, \epsilon$
LRI
ap, ϵ 1
PVARP
as, ϵ 2
S2
vs, ϵ 3
4 $\epsilon, \epsilon_1, \epsilon_2$
URI
5 vp, ϵ
S1

Fig. 2. AVI

The **URI** (Figure 1**(b)**) prevents the pacemaker from simulating the ventricle too quickly thanks to a global clock used to track the time after a ventricular event (VS,VP). URI allows AVI to supply VP only when the global clock is TURI. URI has three interfaces: AVI = $\{\epsilon, \epsilon_1, \epsilon_2\}$, S1 = $\{vp, \epsilon\}$ and S2 = $\{vs, \epsilon\}$ The transitions are:

$$\begin{array}{rr}
\epsilon_1/\epsilon,\text{vs} : 0 \to 1 & \epsilon/\text{vp},\epsilon : 1 \to 0 \\
\epsilon_1/\epsilon,\epsilon : 1 \to 2 & \epsilon_1/\epsilon,\epsilon : 2 \to 3 \\
\ldots & \ldots \\
\epsilon_1/\epsilon,\epsilon : 398 \to 399 & \epsilon_2/\epsilon,\epsilon : 399 \to 0
\end{array}$$

Now we describe the **PVARP** (Figure 3). After an atrial event AS, a ventricular event VS or VP must occur; therfore, when VS or VP is detected, in a small latency time (TPVAB) atrial events are ignored. After TPVAB, there is a second latency time (TPVARP), in which a signal is trasmitted – AR! – outside the pacemakemaker. Finally, after TPVARP, the atrial event can be detected and sent to the LRI component. PVARP has five interfaces: Heart = $\{ar!, \epsilon\}$, LRI = $\{as, \epsilon\}$, AVI = $\{as, vp, \epsilon\}$, S2 = $\{as, vp, \epsilon\}$ and Atrium = $\{as!, \epsilon\}$. The transitions are:

$$\begin{array}{cc}
\epsilon, \epsilon, \epsilon/\epsilon, \epsilon : 0 \to 0 & \epsilon, \epsilon, vp/\epsilon, \epsilon : 0 \to 1 \\
\epsilon, \epsilon, \epsilon/vs, \epsilon\text{: } 0 \to 1 & \epsilon, \epsilon, \epsilon/as, \epsilon : 1 \to 2 \\
\ldots & \ldots \\
ar!, \epsilon, \epsilon/\epsilon, as! : 50 \to 51 & \epsilon, \epsilon, \epsilon/\epsilon, \epsilon : 50 \to 51 \\
\ldots & \ldots \\
\epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon : 51 \to n & \epsilon, as, as/\epsilon, as! : n \to 0
\end{array}$$

Finally, we present **VRP** (Figure 4) which monitors all ventricular events (VP, VS) and filters early events in the ventricular canal that could cause an inappropriate pacemaker behavior. VRP has three interfaces: one named and labeled by our heart system = $\{vs!, \epsilon\}$, and two other interfaces named and labeled S1 = $\{vp, \epsilon\}$ and S2 = $\{vs, \epsilon\}$. The transitions are:

$$\begin{array}{cc}
\epsilon/vp, \epsilon : \text{IDLE} \to 0 & \epsilon/\epsilon, \epsilon : 0 \to 1 \\
\ldots & \ldots \\
\epsilon/\epsilon, \epsilon : 149 \to \text{IDLE} & \epsilon/vp, \epsilon : \text{IDLE} \to \text{s} \\
\epsilon/\epsilon, vs : \text{IDLE} \to \text{s} & vs!/\epsilon, \epsilon : \text{s} \to 0
\end{array}$$

As the reader can notice, CospanSpan(Graph) can model, in a clear and simple way, complex system like the heart-pacemaker system.

5 Lac Operon

In this section we provide a final biological application of the CospanSpan(Graph) algebra: we formalize the Lactose Operon in the Escherichia coli bacterium, using for the first time a natively compositional framework.

The lactose operon in Escherichia coli is composed of a sequence of genes that are responsible for producing three enzymes for lactose degradation, namely the lactose permease, which is incorporated in the membrane of the bacterium and actively transports the sugar into the cell, the beta galactosidase, which splits lactose into glucose and galactose, and the transacetylase, whose role is marginal. The Lac Operon functionality depends on the integration of two different control mechanisms, one mediated by lactose and the other by glucose. Since gene expression is an energy consuming process, Escherichia coli synthesises the proteins involved in the metabolism of lactose when this nutrient is present in the environment and the environment does not provide glucose, which is a more readily available source of energy.

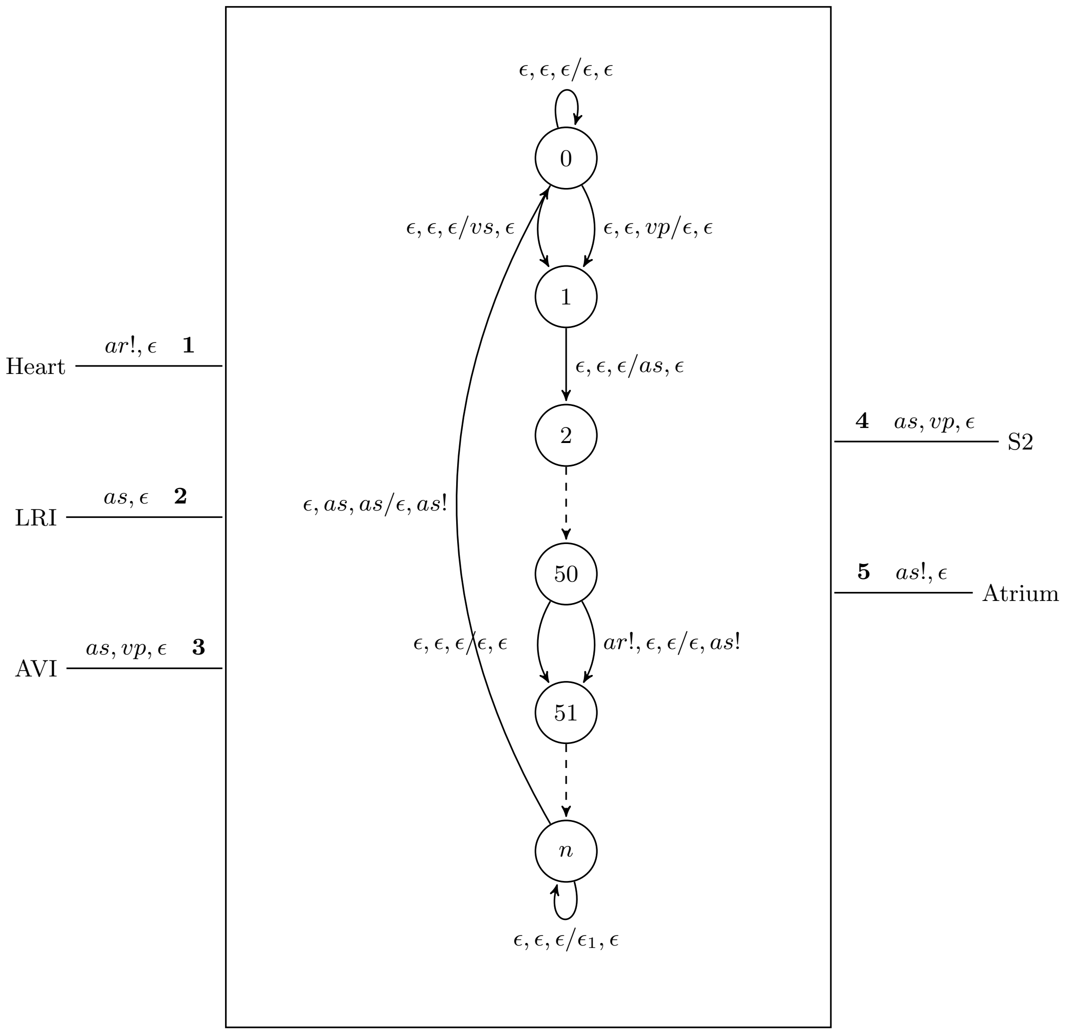

$\epsilon, \epsilon, \epsilon/\epsilon, \epsilon$
0
$\epsilon, \epsilon, \epsilon/vs, \epsilon$
$\epsilon, \epsilon, vp/\epsilon, \epsilon$
1
$\epsilon, \epsilon, \epsilon/as, \epsilon$
2
$\epsilon, as, as/\epsilon, as!$
50
$\epsilon, \epsilon, \epsilon/\epsilon, \epsilon$
$ar!, \epsilon, \epsilon/\epsilon, as!$
51
n
$\epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon$
Heart
$ar!, \epsilon$ **1**
LRI
as, ϵ **2**
AVI
as, vp, ϵ **3**
4 as, vp, ϵ
S2
5 $as!, \epsilon$
Atrium

Fig. 3. PVARP

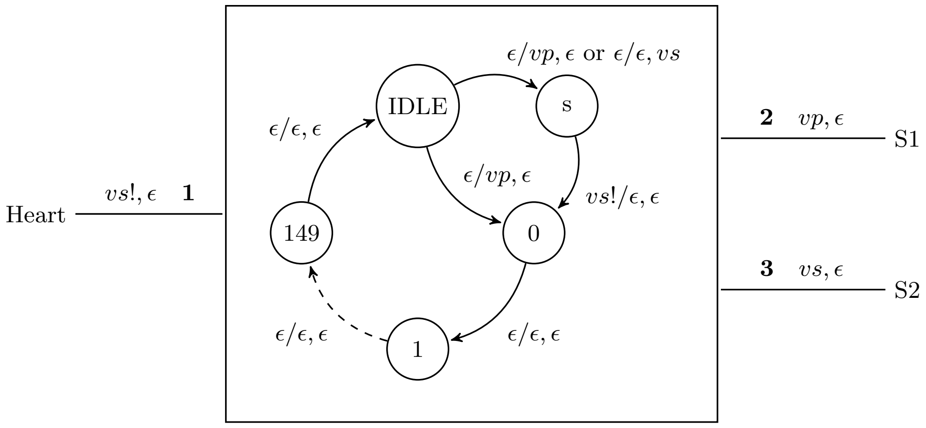

Heart
$vs!, \epsilon$ **1**
2 vp, ϵ S1
3 vs, ϵ S2
IDLE
s
0
1
149
$\epsilon/vp, \epsilon$ or $\epsilon/\epsilon, vs$
$\epsilon/vp, \epsilon$
$vs!/\epsilon, \epsilon$
$\epsilon/\epsilon, \epsilon$
$\epsilon/\epsilon, \epsilon$
$\epsilon/\epsilon, \epsilon$

Fig. 4. VRP

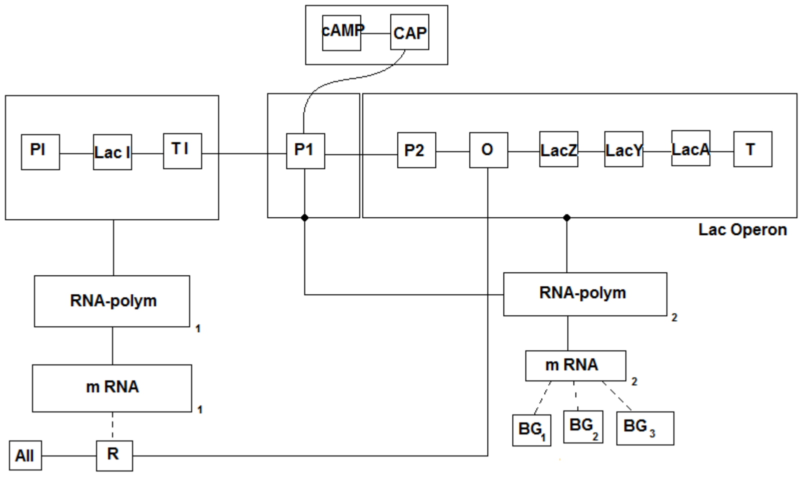
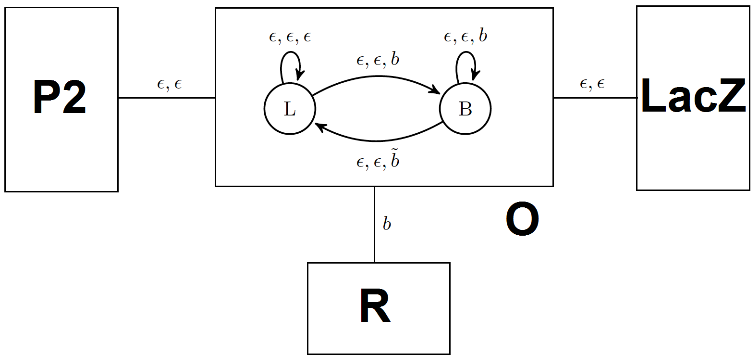


Fig. 5. Lac Operon model

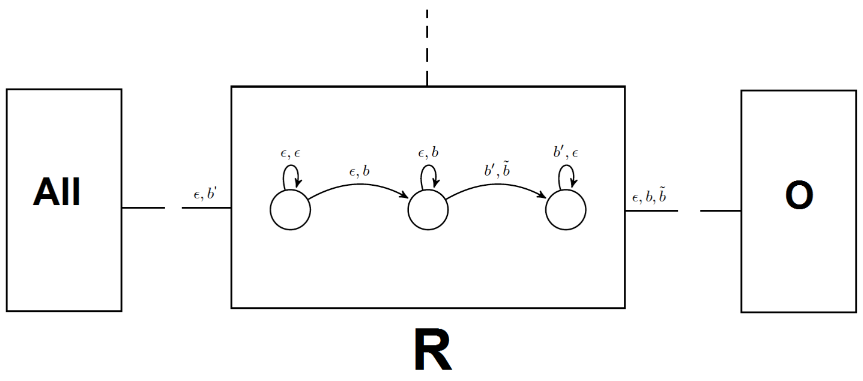
The model, from [5,15,16], that we consider is depicted in graphical form in Figure 5. The DNA sequence of the Lac Operon (depicted in Figure 5) regulates the production of the enzymes, through the genes LacZ, LacY, LacA. The regulation process is as follows: gene LacI encodes the lac repressor R, which, in the absence of lactose, binds to gene O (the operator). Transcription of structural genes into mRNA is performed by the RNA polymerase enzyme, which usually binds to gene P2 (the promoter) and scans the operon from left to right by transcribing the three structural genes LacZ, LacY and LacA into a single mRNA fragment. When the lac repressor R is bound to gene O (that is, the complex R-O is present) it becomes an obstacle for the RNA polymerase, and transcription of the structural genes is not performed. On the other hand, when lactose is present inside the bacterium, it binds to the repressor thus inhibiting the binding of R to O. This inhibition allows the transcription of genes LacZ, LacY, LacA by the RNA polymerase.

A second mechanism is relevant: when glucose is not present, the complex cAMP-CAP, which is present and acting on P1, can increase significantly the expression of lac genes. Of course, also in presence of the cAMP-CAP complex, the expression of the lac genes is inhibited by R-O.

A complete description of the Lac Operon will be provided in a future paper, and it is interesting because the role of the Cospan structure is significant in the modeling. Here, we just give two simple examples, in the following pictures, of the component O:



and of the protein R:



6 Conclusions

In this paper we deeply investigated the compositional feature of CospanSpan(Graph) in modelling biological systems.

A compositional description of these systems is promising because it can provide effective verification techniques. Further developments could be, for example, the integration of time and probability in the description of the Heart-Pacemaker System and a more explicit use of the Cospan structure for modeling the change of behaviors and the creation of new complex components inside a biological system.

References

1. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 1994.
2. J. Bènabou. Introduction to Bicategories. *Reports of the Midwest Category Seminar*, 47:1–77, 1967.
3. D. Besozzi, G. Mauri, G. Păun, and C. Zandron. Gemmating P systems: collapsing hierarchies. *Theor. Comput. Sci.*, 296(2):253–267, 2003.
4. A. Cherubini, N. Sabadini, and R. F. C. Walters. Timing in the Cospan-Span model. *Electr. Notes Theor. Comput. Sci.*, 104:81–97, 2004.
5. L. Corolli, C. Maj, F. Marini, D. Besozzi, and G. Mauri. An excursion in reaction systems: From computer science to biology. *Theoretical Computer Science*, 2012.
6. L. de Francesco Albasini, N. Sabadini, and R. F. C. Walters. The compositional construction of markov processes. *Applied Categorical Structures*, 19(1):425–437, 2011.
7. C. Ferretti, G. Mauri, and C. Zandron, editors. *DNA Computing, 10th International Workshop on DNA Computing, DNA 10, Milan, Italy, June 7-10, 2004, Revised Selected Papers*, volume 3384. Springer, 2005.

8. A. Gianola, S. Kasangian, D. Manicardi, N. Sabadini, F. Schiavio, and S. Tini. CospanSpan(graph): a compositional description of the heart system. *Fundamenta Informaticae*, 171 (1-4):221–237, 2020.
9. A. Gianola, S. Kasangian, D. Manicardi, N. Sabadini, and S. Tini. Compositional modeling of biological systems in CospanSpan(Graph). Technical report, Technical Report, gianola.people.unibz.it, 2020.
10. A. Gianola, S. Kasangian, and N. Sabadini. Cospan/Span(Graph): an algebra for open, reconfigurable automata networks. In *Proc. of CALCO*, pages 2:1–2:17, 2017.
11. Z. Jiang, M. Pajic, S. Moarref, R. Alur, and R. Mangharam. Modeling and verification of a dual chamber implantable pacemaker. In *Proc. of TACAS*, pages 188–203, 2012.
12. P. Katis, N. Sabadini, and R. Walters. A formalization of the IWIM model. In *Proc. of Coordination Languages and Models*, volume 1906 of *LNCS*, pages 267–283. Springer, 2000.
13. P. Katis, N. Sabadini, and R. F. C. Walters. Span(Graph): A categorial algebra of transition systems. In *Proc. of AMAST*, pages 307–321, 1997.
14. C. Martín-Vide, G.Mauri, G. Paun, G. Rozenberg, and A. Salomaa, editors. *Membrane Computing, International Workshop, WMC 2003, Tarragona, Spain, July 17-22, 2003, Revised Papers*, volume 2933 of *Lecture Notes in Computer Science*. Springer, 2004.
15. G. Pardini, R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and S. Tini. Compositional semantics and behavioural equivalences for reaction systems with restriction. *Theoretical Computer Science*, 2014.
16. F. J. Romero-Campero and M. J. Pérez-Jiménez. Modelling gene expression control using P systems: The lac operon, a case study. *Biosyst.*, 2008.
17. N. Sabadini, F. Schiavio, and R. Walters. On the geometry and algebra of networks with state. *Theor. Comput. Sci.*, 64:144–163, 2017.