State Space Exploration of Spatially Organized Populations of Agents

Antoine Dautriche*, Jean-Louis Giavitto†, Hanna Klaudel* and Franck Pommereau*

*IBISC, University of Évry,
523 place des terrasses de l'Agora,
91000 Evry, France
email: [adautriche,klaudel,pommereau]@ibisc.fr

†IRCAM, UMR STMS 9912 CNRS,
1 place Igor Stravinsky,
75004 Paris, France
email: giavitto@ircam.fr

***Abstract*—In this paper, we aim at modeling and analyzing the behavior of a spatial population of agents through an exploration of their state space. Agents are localized on a dynamic graph and they have internal states. They interact with an environment. The evolution of the agents and of the environment is specified by a set of rules. The framework is carefully designed to enable the construction of a global state space that can be automatically build and analyzed.**

The formalism, called *IRNs* for *integrated regulatory networks*, may be seen as an extension of logical regulatory networks (*à la* Thomas) developed in systems biology with spatial information and generalized to use arbitrary data values and update functions of this values. This thus allows to model systems with multiple agents that may be located on a varying spatial structure, may store and update local information, may depend on varying global information and may communicate in their neighborhood. A model of such a system can be defined as an IRN, and then analyzed using model-checking to asses its properties.

This paper sketches the modeling framework and its semantics. We show how IRN may be used for the modeling of a population of simple agents, the automatic analysis of various reachability properties and the use of symmetries to reduce the size of the state space.

I. Introduction

Several dedicated programming language have been developed in Spatial Computing to ease the handling of spatial constraints and resources. However, at the best of our knowledge, they have not been the target of automated analysis like model-checking. Model-checking consists in checking a given property against *every possible execution* of a system model. Such a technique, even if it has limitations, may help the understanding of the complex networks of spatial interactions. In particular, the ability to perform model-checking based analysis makes possible the assessment of causality-related properties and reveal rare events, which usually cannot be obtained through simulation (that can observe only a limited subset of the possible executions).

But this feature constrains the possible choices for modeling. In particular, there should be a finite number of possible system evolutions from a given configuration and they should be enumerable. Moreover, each configuration should be represented in a normalized form, allowing the recognition of two identical configurations. For instance, floating-point positions or attributes are not a possible solution; instead, we shall use solutions based on discrete (qualitative) and combinatorial structures.

This paper aims at proposing such a modeling framework for a population of agent thought as a *spatially distributed dynamical system*. We are particularly interested by the modeling of the dynamics of space: either because the agents move or because the spatial organization is dynamic. The model integrates the behavior of autonomous agents with their spatial behavior: their location and their interaction. It takes into account variables specifying information about the agents (their internal states) as well as their spatial organization and the environment (global variables). The evolution proceeds by the asynchronous update of a variable.

The resulting framework, called *IRNs* for *Integrated Regulatory Networks*, is illustrated on a simple example of mobile agents. It was initially motivated by the modeling of growing tissues at a cellular level in systems biology and targeted applications like developmental processes, invasive cancers, plant growth, etc. In these application domains, agents are cells and the behavior of an agent is controlled by a genetic regulatory network. One of the motivations of this paper is to show that the techniques developed for this application domain can be of a wider use (in particular, they are not limited to model regulatory processes).

The formalism proposed here encompasses the *generalized logical formalism* initially proposed by René Thomas in the seventies [1], [2] by considering arbitrary finite domains for the variables and arbitrary functions between them to model the evolution of the variables. It extends the proposal we made in [3] to integrate the handling of spatial relationships, by: *1)* including a more general notion of observables and a more orthogonal management of variables and measures, *2)* considering general labeled graph to model the spatial relationships, and *3)* including the handling of environmental factors which are not necessarily under the direct control of the agent (*e.g.*, a

temperature) but affect the model dynamics (*e.g.*, a rate of diffusion).

II. Integrated regulatory networks

A. Intuitive presentation

The notion of IRN is presented in several steps. First, we consider a population of agents. Each agent is characterized by an internal state. For the sake of simplicity, we assume here that agents are homogeneous and exhibit the same behavior. (This is not a limitation because one can use a variable whose value allows to switch between various distinct behaviors.) In [3], [4] agents are called *modules*.

The internal state of an agent is defined as a set of *local variables*. Each local variable *lvar* is associated with an *update function* with the same name that provides the following information: its codomain defines the range of values *lvar* can assume, the current value of *lvar* being denoted as x_{lvar}; its arguments define the variables (and measures, cf. below) *lvar* depends on.

The evolution of a local variable within an agent may depend on the values of some local variables in other agents. This is modeled as a *local measure*, allowing to collect those values in the neighborhood of the evolving agent. For example, a local measure may integrate the concentrations of a regulatory component diffused toward a cell from its neighbors according to the distances between cells.

Next, to model spatial relations between agents, we *localize* them on a *topological collection*. Topological collections have been introduced in [5] to describe arbitrary complex spatial structures that appear in biological systems [6] and other dynamical systems with a time varying structure [7], [8]. For the simplicity of the presentation, we consider in this paper that the neighborhood relationships between agents are represented by a *partially labeled graph*. For example, the graph may reflect the spatial arrangement of cells in a tissue, or the positions of agents in a building. Some vertexes of this graph are agent identifiers and each such vertex and its associated label corresponds to a given agent. The other vertexes are "empty locations" (places in space where there is no agent). They do not have labels. Two agents i and j are neighbors if there is an edge between the vertexes labeled i and j.

Topological collections are used to implement a database that records the neighborhood relationships, which can be queried and updated efficiently. So, local variables become bindings attached to the vertexes of the collection. Similarly, local measures taken from an agent i become computations of a single value from (possibly several) multisets of pairs $(\ell_j, x_{lvar\,j})$ where ℓ_j is the label of the arc between i and j and $x_{lvar\,j}$ is the value of a local variable *lvar* of an agent j.

The whole graph may be labeled itself, yielding *global variables*. Global variables may reflect environmental parameters like temperature or pressure. Moreover, *global measures* are obtained by computing a value from an observation of the whole graph. For example, a global measure may be the number of cells in the tissue.

Finally, we introduce *graph updates* to allow for evolutions of the structure of the graph. They correspond to spatial modifications in the system, like agents movements, creation or the disappearance of agents, but also transformations of the underlying space. However, we make the assumption that a graph update is triggered from some agent. Note that this is not a limitation because a special agent may be used to model the environment, *i.e.*, to trigger "spontaneous" changes.

The dependencies between variables, measures and graph are subject to some constraints which are explained below.

B. A Graphical Syntax

To support intuition, IRNs are depicted using a graphical notation inspired by that largely used to describe regulatory networks in systems biology. Beware not to confuse the graph of an IRN with the graph used to represent the spatial relationships.

In the graphical representation of an IRN, the vertexes are variables or measures and an arc from one variable *A* to a variable *B* indicates that the value of *B* depends on the value of *A*. The graphical conventions are described in Figure 1. The shape and style (dotted or plain line) of the components denote their nature while the arcs indicate a potential influence (resulting in corresponding arguments in the function). Note that influences may be arbitrary and are not limited to inhibition or activation, as in logical regulatory networks.

The constraints about the arcs are defined consistently with the arguments allowed for each kind of update function. The rationale is as follows: local variables and measures can depend both on local and global information that is available to every agent; global variables or measures

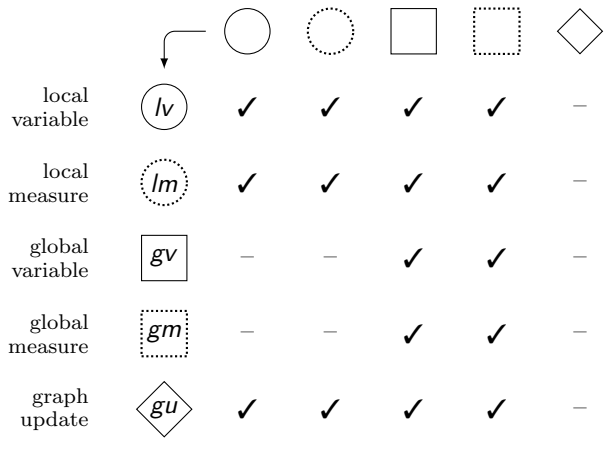

Fig. 1. Graphical conventions: measures are depicted in dotted lines; local objects are depicted by round nodes while global objects are depicted by square nodes. The check marks indicate whether an arc is allowed (✓) or not (–) from each node type in the top row toward the node types in the left column.

occur globally in the system and so cannot depend on any local information; graph updates occur at some given agent and thus can depend on local or global information, but no function can depend on a graph update because it does not compute any value but instead transforms the spatial structure. Moreover, we assume that any variable may depend on its current value, so we do not need to draw self-loops in the graphical representation.

Finally, we require that an IRN is *well-formed*, in the sense that there is no mutual recursion between measures (cycle between measure nodes) and all functions are total on their finite domain and computable.

C. The Little Horses Game

In the rest of this paper, we illustrate the use of the IRN formalism with the modeling and the analysis of a modified version of a children game called *The Little Horses*. We consider this example for its pedagogical value. It does not involve any transformation of space but only agent destructions and movements, which is enough to illustrate the expressiveness of our approach.

This game is played on a board. We consider here a one-directional discrete ring (a finite set of slots) on which the horses (the agents) move following a predefined direction. The behavior of a horse is specified by a reactive systems similar to the *animat* approach developed by P. Maes *et al.* [9]. Each horse evolves according to the IRN depicted in the right of Fig. 2:

- E is a local variable representing the energy level of the horse, ranging in $\{0, 1, 2\}$.
- A is a local variable representing the appetite of the horse, ranging in $\{\mathsf{true}, \mathsf{false}\}$. It is hungry ($x_A = \mathsf{true}$) when its energy is below 2 and then it is kept busy eating. The horse stops to be hungry ($x_A = \mathsf{false}$) when its energy level reach 2.
- F is a global measure whose value indicates the occupation of the next two slots in the ring. F is global instead of local because a local measure queries the variables of the agents in the neighborhood of another one. Here we are not interested in these values, but only in the presence of agents.
- J is a graph update that represents a jump of the horse i from a slot to a next one. To do so, the local state must be such that $x_E = 2$ (enough energy) and $x_A = false$ (not busy eating), then there are several cases:
  - if the next slot is free, the horse simply jumps there and its energy is dropped to $x_E = 1$;
  - if the next slot is occupied by a horse j whose energy is 0, then i is allowed to jump where j is currently located; then j is removed from the board and the energy of i is dropped to $x_E = 1$.
  - if the next slot is occupied by a horse j whose energy is not 0 and the slot next to j is free, then i is allowed to jump toward the slot next to j and its energy is dropped to $x_E = 0$;
  - in any other case, graph update J cannot occur.

  To obtain information about the surrounding in order to perform J or not, the horse needs the global measure F to know which one among the two next slots is free (if any) and it needs the local measure $E1$, allowing to sense the energy of a horse in the next slot.

The corresponding IRN is graphically depicted in Fig. 2 (left and right). There is no global variable in this model. In some version of the game, the direction of the movement changes each time a horse is removed. An example of possible global variable would be the direction of the movements.

D. Formal specification of the framework

The IRN is specified by the finite set of its global and local variables with their corresponding variable update functions, its local and global measures definitions, and its graph update functions. A *state of an IRN* is defined by a binding λ of its global variables and a topological collection C that will define, in particular, the bindings for the vertexes (local variables of the agent located on this vertex) and the edges.

1) Variables, Measures and Update Functions: We assume that IRN *variables* range over finite sets of values and may be updated using *variable update functions*. For each *global* or *local* variable var, there is a unique update function that computes the new value of var and whose allowed parameters are defined consistently with the constraints given in Figure 1 (*e.g.*, a global variable update may take as parameters only global variables or global measures as shown in the corresponding row).

A local variable has a value in each agent i in C, which is computed by an update function with the same name (common to all agents). For example, in the Little Horses modeling, local variables A and E may be specified by giving their update functions:

$$A(x_{Ei}) \in \{\mathsf{false}, \mathsf{true}\}$$
$$x_{Ei} \longmapsto (x_{Ei} < 2)$$
$$E(x_{Ei}, x_{Ai}) \in \{0, 1, 2\}$$
$$x_{Ei}, x_{Ai} \longmapsto \text{if } x_{Ai} \wedge (x_{Ei} < 2) \text{ then } x_{Ei} + 1 \text{ else } x_{Ei}$$

where x_{Ai} is the value of the local variable A for agent i.

Measures are obtained from observations of the topological collection C. We distinguish *global measures* that are obtained by observing C globally, and *local measures* that are obtained by observing an agent i and its neighborhood in C.

A global measure is defined by a function (with the same name) that returns a value in a finite set, taking as parameters C and the binding λ for the global variables, and possibly other parameters as allowed by Figure 1. For example, in the Little Horses modeling, global measure F has the following signature:

$$F(C, \lambda, i) \in \{\mathsf{Slot1}, \mathsf{Slot2}, \mathsf{NoSlot}\}$$

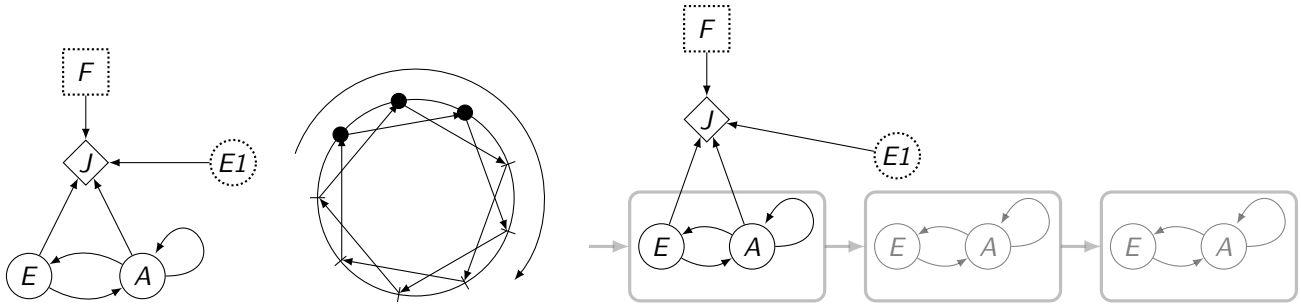

Fig. 2. *Left:* The IRN that describe the behavior of a horse. *Middle:* A possible spatial organisation of the Little Horses board, with a 9-slots ring and 3 consecutive horses on it. Note that each slot has two neighbors, the next slot on the ring and the second next. *Right:* Graphical representation of the IRN specification of Little Horses; in the upper level, global measure F, local measure $E1$ and graph update J; in the lover level, local variables A and E; in gray, a partial view of a the ring C with three occupied vertexes (so we represent the corresponding local variables; note that when a slot is empty, there is no associated local variables: local variables are attached to agent not to vertexes).

and returns Slot1 if the slot next to the slot i is free, Slot2 if the slot $i+1$ next to i is occupied and the slot $i+2$ is free and NoSlot elsewhere.

A local measure *lm* is defined using a function with the same name that returns a value from a finite set, taking as parameters the values of *lvar* in all the neighbors of i for each local variable *lvar* depends on, and possibly other parameters as allowed by Figure 1. These values for each such *lvar*, denoted by $\overline{x_{lvar@i}}$, are collected as a multiset of pairs $(\ell_j, x_{lvar\,j})$, where ℓ_j is the label of the edge between i and j in C. In the Little Horse example local measure E may be specified as:

$$E1(\overline{x_{E@i}}) \in \{0,1,2\} : \overline{x_{E@i}} \longmapsto x_{E(i+1)}$$

returning the energy level of the neighbor (if this has a meaning, that is, if the slot next to i is occupied by a horse). Here the function simply selects an element in the multiset. But using a multiset to collect the local variables of the agent in the neighborhood enables the computation of aggregate measures, even if the neighborhood is non-uniform or dynamic.

A *graph update* is a function that takes as parameters a topological collection C, a binding λ for global variables, and an agent identifier i. It either returns an empty set when the application conditions have not been met, or computes a set of new collections $\{C_1, \ldots, C_n\}$ $(n > 0)$. In terms of topological collections, a graph update corresponds to a set of transformations.

2) State of an IRN and Dynamics: A *state* of an IRN is represented by a pair (λ, C) where λ is a binding assigning values to the global variables, and C is a topological collection that records the graph structure and the value of the local variables as bindings attached to the vertexes. A topological collection is a function from vertexes and edges to values. The local variable of an agent are gathered in a record. In other words, we have $x_{gvar} = \lambda(gvar)$ for a global variable *gvar* and $x_{lvar\,i} = C(i).lvar$ for a local variable *lvar* and a vertex i.

Schematically, a state (λ, C) of an IRN may evolve to another one (λ', C') in one of the following manners, if the corresponding application conditions are met and if $(\lambda', C') \neq (\lambda, C)$:

- by applying a graph update *gup* to the current topological collection for an agent i: the resulting topological collection may be any C' returned by $gup(C, i)$, and the global variables are unchanged $\lambda' \stackrel{\text{df}}{=} \lambda$;
- by applying a global variable update: the resulting topological collection is unchanged $C' \stackrel{\text{df}}{=} C$, and binding λ' is λ updated for some global variable *gvar* such that $\lambda'(gvar) \stackrel{\text{df}}{=} gvar(\cdots)$;
- by applying a local variable update on an agent i of C: the resulting topological collection C' is C where *lvar* at agent i has been updated, *i.e.*, $C'(i)(lvar) \stackrel{\text{df}}{=} lvar(\cdots)$ and $C'(i')(v) \stackrel{\text{df}}{=} C(i')(v)$ for every $(i', v) \neq (i, lvar)$, and the global variables are unchanged $\lambda' \stackrel{\text{df}}{=} \lambda$.

Given an initial state (λ_0, C_0), and the evolution rules defined above, one may build a transition system $\longrightarrow$ describing the dynamics of the IRN.

III. Building an IRN State Space and Checking its Properties

We have implemented a tool for the automatic building of the state graph reachable from a given state according to the IRN dynamics. This prototype allows an exhaustive exploration of the state space of the IRN.

This tool is composed of two parts: one part is the coding in MGS of all the update functions. MGS [7], [10] is an experimental spatial programming language implementing the concepts of topological collections and their transformations. For example, the rule:

```
{appetite=0, energy=2} as b, u:emptyslot
/ (^b == i)
⟶ u, b+{energy=1};
```

specify one case of the graph update J. The rule takes the abstract form $x, y/cond \to exp$. The left hand side matches two labeled vertexes x and y in the graph that are linked by and edge (this is specified by the comma between x and y). If they satisfy the condition *cond*, the labels are replaced

by the labels computed in the right hand side. The pattern `{appetite=0, energy=2} as b` matches a record with at least two fields `appetite` and `energy` respectively with value 0 and 2. The entire record is referred as b. The label referred by u must be of a predefined type `emptyslot` representing an empty location. The expression `^b` denotes the vertex labeled by `b`. So the entire pattern specify a configuration where a horse with enough energy can jump to the next slot which is empty. The right hand side describes the labels resulting from this jump. The notation `r + r'` denotes the asymmetric merge of records: the fields of `r + r'` are the fields of `r` and `r'` where the value of the field of `r` have been updated by those of `r'`.

The second part of the tool is a driver (written in Python) that triggers the computation of all the states s' satisfying the relation $s \longrightarrow s'$ for a given state s. By iterating this procedure, we can build the *orbit* of a state (the states s' such that $s \rightarrow^* s'$) and more generally, the graph G corresponding to the transition system. The driver is generic: it takes the signature of the update functions of an IRN and the corresponding MGS implementation and builds the states reachable from a given one.

Several reachability properties can be checked using standard graph algorithms on G. A state s' is reachable from a given state s if there is path from s to s' in G. A *steady state* is a state without output edges. A *garden of Eden* is a partition of G in two components with no edge going from the second component to the first (the garden of Eden can only be leaved). The strongly connected components with no output edges, are the *bassins of attraction* of the system. If the transition system is deterministic they are also the limit cycles of the system, etc.

IV. Reducing the State Space Size using Symmetries

One main shortcoming of this approach is the (unavoidable) explosion of the state space. Several techniques can be used to limit this explosion. We sketch here a method relying on the detection of symmetric executions.

An equivalence relation $\simeq$ is a bisimulation for the relation $\longrightarrow$ iff $(s_1 \simeq s_2)$ and $s_1 \rightarrow s_1'$ imply that there exists s_2' such that $s_2 \rightarrow s_2'$ and $s_1' \simeq s_2'$. If $\simeq$ is a bisimulation for $\longrightarrow$, then reachability properties can be checked on the reduced system $\longrightarrow / \simeq$.

Here we consider equivalence relations triggered by automorphism of topological collection (that is, in the context of this paper, automorphism of labeled graphs). For example, for the Little Horses, it is intuitive that configurations that are obtained by a rotation of the ring must be equivalent (they have all the same future up to a rotation). Checking for graph automorphism is an heavy task, but note that the automorphism applies on the board (which has a limited size), not on the state space (which grows exponentially with the size of the board).

We have devised a method to check "on the fly" that two states are equivalent. When a new state s' is built, the driver checks that it does not exists already an equivalent state s in the current reachability graph G. This is done by first converting s' into a graph $g_{s'}$ and then checking the existence of an automorphism with the graph g_s (the existence test is implemented in MGS using the NAUTY library [11]). The graph g_s associated to a state s corresponds to the graph of the underlying topological collection, labeled consistently with the values of local variables. Fig. 3 shows how handling symmetries reduces the size of the state space for various instances of the Little Horses example.

agents	states		transitions		stable
	full	reduced	full	reduced	states
1	21	5	21	5	0
2	199	28	358	55	0
3	981	214	2446	554	0
4	2828	526	8638	1669	0
5	4345	629	14616	2220	1

Fig. 3. Number of states and transitions in the full and reduced state-space of the Little Horses IRN, depending on the number of agents in the system (for a fixed ring of 5 slots). Full and reduced state-spaces have the same number of stable states. When there are 5 agents, a stable state is found, corresponding to the case where no move is possible. In every other case, there always exists at least one free slot so that moves are always possible. In this latter situation, there exists attractors in the system, which can be checked by searching the terminal strongly connected components of the state-space.

V. Related and future works

In this paper, we advocate the use of automated model-checking techniques for a class of multi-agent systems taking into account dynamic spatial relationships. Notice in particular that in our proposal, communication is possible through two ways: on the one hand, data local to each agent may be "sensed" from its neighborhood; on the other hand, agents may trigger transformations of the spatial structure around them. In the latter case, and agent may relabel its attached edges to "send" information to its neighbors.

We have also presented preliminary results for the automated building of the state space and we have illustrated the approach on a toy example, showing in particular the benefits of symmetries reduction.

Discrete algebraic formalisms like process algebras or Petri nets are relevant for the modeling of multi-agent systems because automated tools can be used to help both the modeling and the systematic analysis of system behaviors. Some process algebras (*e.g.*, used to study mobility or variants of π-calculus used for biological modeling) include a notion of localization but often the spatial relationships are not explicitly exposed (the algebra of locations is encoded into identifiers) or too limited (nesting structures). The IRN framework presented in this paper

relies on topological collections and topological collection rewriting to manage spatial information. This approach has been validated in many applications and for a wide variety of spatial representations. The MGS language is used to implement the underlying computation under the supervision of a generic driver. A complete description of the driver is given in [12]. So, with respect to other formalisms, IRNs provide the modeler with an explicit and powerful tool to represent and manipulate spatial information. This is typically strongly either abstracted (see, *e.g.*, "membranes" in P-systems [13]) or captured by a geometrical approach (see, *e.g.*, [14], [15] for surveys) which is not suitable for model-checking.

The IRN formalism may be enriched and extended in several directions. Natural extensions to many different kinds of agents may easily be provided. Future works will provide a generalized formal definition of the framework presented in the paper. We also intend to run case studies in order to assess the relevance of our proposal. In particular, running larger examples will allow to assess the scalability of the method with respect to model size and number of agents, as well as the efficiency of symmetry reductions to reduce state space explosion.

Concerning spatial transformations, we focus here on labeled graphs. However, the notions of topological collection and topological rewriting are more general and may handle higher dimensional objects, a feature relevant in a lot of application areas [16].

Another direction of future research consists in relaxing some constraints concerning the definition of the dynamics; for example, to allow alternative update strategies or infinite state spaces. Concerning the former, the current framework defines the dynamics of the system using an asynchronous strategy. This approach is relevant, *e.g.*, for regulation networks but may be cumbersome in other application domains. The design space of update strategy is largely open, from asynchronous to synchronous and from deterministic to non-deterministic ones. In particular, it is usually observed that using a synchronous update strategy results in smaller state-spaces (but with states that are not reachable using asynchronous updates), which may be another direction to achieve better scalability. For the latter research direction, the actual IRN specification restrictions ensure finiteness of the state space, but may become artificial in practice. Some of these restrictions may be relaxed by resorting to abstraction and specific reduction techniques, like those in [17], [18], which originally have been designed for systems with dynamic process creation.

Acknowledgements: This work is partially supported by the ANR research projects AutoChem, Calamar and SynBioTIC.

References

[1] R. Thomas, "Boolean formalization of genetic control circuits," *J. Theor. Biol.*, vol. 42, pp. 563–85, 1973.

[2] R. Thomas, D. Thieffry, and M. Kaufman, "Dynamical behaviour of biological regulatory networks–I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state." *Bull. Math. Biol.*, vol. 57, pp. 247–76, 1995.

[3] J.-L. Giavitto, H. Klaudel, and F. Pommereau, "Qualitative modelling and analysis of regulations in multi-cellular systems using petri nets and topological collections," in *Proceedings Fourth Workshop on Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC 2010)*, vol. 40. Electronic Proceedings in Theoretical Computer Science (EPTCS), 23-24 August 2010, arXiv e-prints 1011.0498.

[4] C. Chaouiya, H. Klaudel, and F. Pommereau, *A modular, qualitative modelling of regulatory networks using Petri nets*. Springer, 2010, ch. 12 of *Modeling in Systems Biology – the Petri Net Approach*.

[5] J.-L. Giavitto and O. Michel, "The topological structures of membrane computing," *Fundamenta Informaticae*, vol. 49, pp. 107–129, 2002.

[6] ——, "Modeling the topological organization of cellular processes," *BioSystems*, vol. 70, pp. 149–163, 2003.

[7] J.-L. Giavitto, "Topological collections, transformations and their application to the modeling and the simulation of dynamical systems," in *14th International Conference on Rewriting Technics and Applications (RTA'03)*, ser. LNCS, vol. 2706. Valencia: Springer, Jun. 2003, pp. 208–233.

[8] J.-L. Giavitto and A. Spicher, "Topological rewriting and the geometrization of programming," *Physica D: Nonlinear Phenomena*, vol. 237, no. 9, pp. 1302 – 1314, 2008, novel Computing Paradigms: Quo Vadis?

[9] P. Maes, "A bottom-up mechanism for behavior selection in an artificial creature," in *proceedings of the first international conference on simulation of adaptative behavior*, B. Book, Ed. MIT Press, 1991.

[10] O. Michel, J.-L. Giavitto, J. Cohen, and A. Spicher, "The MGS homepage," http://mgs.spatial-computing.org.

[11] B. D. McKay, "Nauty version 2.2," http://cs.anu.edu.au/people/bdm/nauty/, 1994–2003.

[12] A. Dautriche, "Construction de l'espace des états d'un IRN et techniques de réduction," Master's thesis, University of Evry, Master MOPS, Jul. 2011, (in French).

[13] G. Paun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, 2000.

[14] A. T. Bittig and A. M. Uhrmacher, "Spatial modeling in cell biology at multiple levels," in *Winter Simulation Conference*, 2010, pp. 608–619.

[15] K. Takahashi, S. N. V. Arjunan, and M. Tomita, "Space in systems biology of signaling pathways - towards intracellular molecular crowding in silico," *FEBS Letters*, vol. 579, no. 8, pp. 1783 – 1788, 2005.

[16] E. Tonti, "On the mathematical structure of a large class of physicial theories," *Rendidiconti della Academia Nazionale dei Lincei*, vol. 52, no. fasc. 1, pp. 48–56, Jan. 1972, scienze fisiche, matematiche et naturali, Serie VIII.

[17] H. Klaudel, M. Koutny, E. Pelz, and F. Pommereau, "An approach to state space reduction for systems with dynamic process creation," in *ISCIS'09*, ser. IEEE digital library. IEEE, 2009.

[18] ——, "State space reduction for dynamic process creation," *SACS*, vol. 20, 2010.