Model-Checking an Ecosystem Model for Decision-Aid

Marie-Odile Cordier, Christine Largouët, Yulong Zhao

To cite this version:

Marie-Odile Cordier, Christine Largouët, Yulong Zhao. Model-Checking an Ecosystem Model for Decision-Aid. IEEE 26th International Conference on Tools with Artificial Intelligence, Nov 2014, Limassol, Cyprus. <10.1109/ICTAI.2014.87>. <hal-01094742>

HAL Id: hal-01094742

https://hal.archives-ouvertes.fr/hal-01094742

Submitted on 16 Dec 2014

Model-Checking an Ecosystem Model for Decision-Aid

Marie-Odile Cordier
University of Rennes 1
IRISA-UMR6074
Rennes, France
Email: cordier@irisa.fr

Christine Largouët
Agrocampus Ouest
IRISA-UMR6074
Rennes, France
Email: clargoue@irisa.fr

Yulong Zhao
University of Rennes 1
IRISA-UMR6074
Rennes, France

Abstract—**This work stems on the idea that timed automata models and model-checking techniques may bring much in a decision-aid context when dealing with large and interacting qualitative models. In this paper, we focus on two key issues when facing the interpretation and explanation of behavior in real-world systems: the model building and its exploration using logic patterns. We illustrate this approach in the ecological domain with the modeling and exploration of a fisheries ecosystem.**

I. Introduction

In numerous application domains, the vast majority of models are simulation ones. However, as model complexity increases, it becomes more and more difficult to simulate real systems without being exposed to the problem of scale. Moreover, in many cases, appropriate data and knowledge are not available to supply real-valued models. At the same time, a very attractive approach, model-checking, has been introduced in computer science and is now widely applied to verify if the behavior of a timed system is complying with its specifications. Our work stems on the idea that model-checking could help a lot for the decision-aid task in large and complex systems. It is especially true when we want to explore the impact of a decision on real-world systems.

Classical model-checking techniques are dedicated to finite state systems. However many systems are usually represented by analytical models as a set of differential equations. Some studies present how to quantize continuous-time systems as discrete-event systems in order to diagnose them [1]. Some others, in the biological and biomedical field, promote qualitative models to analyze complex and large systems in a formalism that is closed to finite-state systems [2][3]. Qualitative modeling [4][5] relies on solid theoretical foundations to provide a reliable abstraction of real-world models. Nevertheless, even for qualitative models, the exploration still remains costly and usually relies on exhaustive simulation. Recent studies have emphasized the great interest of coupling qualitative models, generally acquired from expert knowledge, with model-checking techniques [6][7][8]. But writing a temporal logic request remains a challenging step reserved to experts. Query patterns have already been defined for the verification of industrial applications [9] or biological systems [10]. However translating requests that help the user understanding the possible system trajectories, when applying new decisions, has not been yet studied.

This paper presents two original contributions in the use of model-checking techniques for decision-aid in large and complex systems. Firstly, we propose a method to build automatically a network of timed automata, tractable in practice for decision problems. The key point is to quantize the continuous-time sub-systems and to automatically get a network of timed automata from an abstracted description of the system. The originality of this approach is to reduce the model complexity, which is important after the automatic generation, by using a hierarchical classification algorithm. Secondly, we propose high-level query patterns to explore and predict future changes in a decision-aid context. We illustrate our approach in the field of ecology with the modeling and exploration of fisheries ecosystems. The global ecosystem is defined as a set of interacting components such as the anthropogenic pressures, the environmental units and the continuous predator-prey subsystems.

This paper is organized as follows. Section 2 describes our illustrative ecosystem example. Section 3 gives more details on the qualitative model described as a timed automata network. Section 4 explains the model building algorithm and details the reduction performed. Section 5 focuses on the definition of high-level logic patterns well-suited for the model exploration by non-specialists. Section 6 presents three kinds of results: performance tests on benchmark models, the EcoMata software and finally the application on real-world applications such as a lagoon fisheries system in New-Caledonia. Some elements of conclusion are given Section 7.

II. Illustrative example

A theoretical ecosystem of interacting species under fishing pressures and climatic events is presented Figure 1. The system is composed of three units, each of which structured in components. These three units interact through potentially complex ways, via synchronized events and timing constraints. The components of *ENV* and *AP* are naturally expressed in a qualitative way while *ES* is usually represented by a population dynamics model.

- *AP* contains two components: the two fishing pressures *PP0* and *PP1* related to the exploitation of the fish species *SP0* and *SP1*.

- *ENV* contains the *Hurricane* and *Warming sea* components that impact respectively the species *SP2* and *SP3*.
- *ES* models the dynamics of the ecosystem itself, the fish species having predator-prey interactions. *ES* contains four components: the four fish species *SP0* to *SP3*. The species *SP0* is the predator of species *SP1* (the biomass flow is represented by an arrow) while species *SP2* and *SP3* are the preys of *SP1*. These components follow a population dynamics modeling such as Lotka-Volterra equations [11].

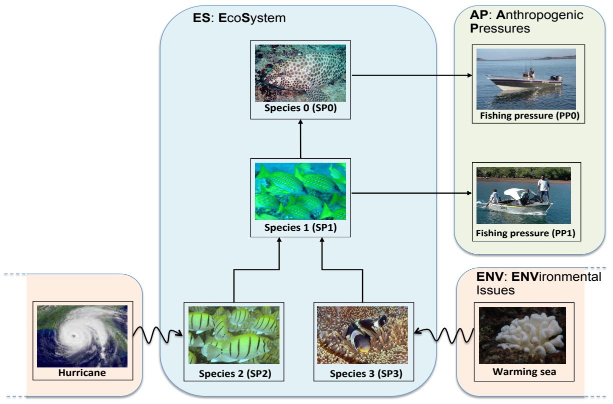

Fig. 1. Interacting species under fishing and environmental pressures

III. Timed Automata Network

We are interested in systems where time constraints have a key role and should be dealt with in an explicit way. The discrete-event representation given by timed automata is an appropriate formalism to model this kind of systems. The system states of each component can be represented using qualitative levels and expressed using a property associated to locations. For our illustrative example, the biomass values as well as the fishing pressures are described using qualitative levels *Low*, *Medium*, *High* and *Endanger*. For the *ENV* components, we can distinguish two states: before and after the climatic event (with a timing constraint for the *WarmingSea*).

A Timed Automaton (TA), first proposed by Alur and Dill [12], is a finite automaton (a graph containing a finite set of locations and a finite set of labeled edges) extended with real-valued variables. The variables model the logical clocks in the system, that are initialized with zero when the system is started, and then increase synchronously with the same rate. Clock constraints are of two types: *invariants* which restrict the way time may elapse in a location and *guards* which restrict the temporal occurrence of a transition. Clocks may be reset to zero when a transition is taken.

The system model is then defined as a *network of timed automata* having as many timed automata as components. There is only one clock defined for each automaton. The numerical constraints, the guards and invariants, express the dynamics of the system and are obtained from user requirements or from any analytical dynamic model that describes the system.

Synchronized events model efficiently the interaction between components and local clocks allow each component to have its own temporal scale. Each component is represented by one automaton and the different automata are synchronized through the shared event labels of the edges. This means that edges of different automata labeled with the same synchronized event are taken simultaneously. The behavior of the global system is obtained by synchronizing the timed automata on the event labels they share, this operation is called *parallel composition* [13].

The network of timed automata (NTA) is the finite family $A_{i\{1\leq i\leq n\}}$ such that n is the number of components. For each A_i of a NTA, we define the set of *neighbouring timed automata* $Nei(A_i) = \{A_j\}_{\{1\leq j\leq n-1\}}$ as the set of automata A_j such that A_j shares at least one synchronized event with A_i (A_i having at most $n-1$ neighbours). To model the system dynamics, there are two kinds of locations in an automaton A_i:

- *Stable locations* refer to states associated to qualitative properties that may only evolve when receiving synchronized event (stable locations are also called qualitative states). If no event is received, the system stays in his stable location. For instance, the TA of *SP1* contains four stable locations *sp1_H, sp1_N*, *sp1_L* and *sp1_E* corresponding to the four biomass qualitative levels *High*, *Normal*, *Low* and *Endanger*. It is quite similar for the fishing pressure automata, for which we can denote three stable locations for the three qualitative fishing pressure levels: *High*, *Normal* and *Low*
- *Transient locations* correspond to an increasing or decreasing dynamics of a qualitative level. Transient locations correspond to a period of time in which the subsystem is evolving between two stable locations. The invariants express the maximum delay of stay in this transitory location.

Let A be a NTA of n automata A_i, each one being defined by N qualitative levels. Any possible change of qualitative state in $Nei(A_i)$ may trigger a change in A_i and then determines a transient location. If we consider that each A_i can have at most j neighbouring automata, the maximum number of transient locations of each A_i is then N^j.

The edges of A_i are of four types:

- Edges whose source is a stable location and destination a transient one. They are labelled by the triggerring event received from a neighbouring automaton. The local clock is reset and is used to express the time elapsed in the transient location.
- Edges whose source is a transient location and destination a stable location. The minimum duration allowed in the source transient location is expressed by the guard. This kind of edges correspond to a move to a new stable state. An event specifying the type of change is emitted.
- Edges whose source is a transient location and destination the stable location from which the evolution has been initiated. This edge is a ”return edge” and cancel any evolution when a contradictory event occurs from a

neighbour.

- Edges between transient locations. When a new event happens during an evolution, it could be necessary to adjust the temporal constraints to increase or decrease the evolution speed and then the time spent in the transient location.

Let us illustrate the modeling process for an ecosystem of only three species $SP0 \rightarrow SP1 \rightarrow SP2$ where *SP0* is the prey of *SP1* and *SP2* the predator of *SP1*. Figure 2 shows part of the *SP1* species automata. Stable locations are represented in white circles while transient locations are in black. The local clock is *t*. On this simplified automaton *SP1* can move from its initial location *sp1_N* to a transient location when receiving one of these three synchronized events *sp0_H?*, *sp2_N?* or *sp2_L?*. The received event as well as the qualitative level of the other neighbors determine the triggered edge towards the transient location. For instance when the prey *SP0* has increased to the *High* level and emitted an *sp0_H!* event, the duration of the *SP1* increase depends also on the level value of its predator *SP2*. When the predator *SP2* is in a low level (*sp2==L*) *SP1* will increase more quickly to the high level than if *SP2* is in a normal level (*sp2==N*). Consequently two transient locations need to be designed. They differ from their temporal constraints: the invariant and the guard on the outgoing edge. In Figure 2, the right transient location represents a faster increase of the *SP1* biomass level for a same received event *sp0_H?*. Using this right path, it takes from 23 to 36 time units to get to the location $sp1\_H$. If a neighbor automata evolves during the increasing step, for instance *sp2_N?* is received while the system is already in a transient location, the system moves to a another transient location and computes a new clock value taking into account the time already elapsed in its former transient location and the new increase speed. When *SP1* finally moves to the stable location *sp1_H*, a synchronized event *sp1_H!* is emitted and its neighbor automata will evolve in turn. Figure 2 is only a part of the $SP1$ species automaton, the corresponding decreasing part exists between the *sp1_H* and *sp1_N* locations, the both increasing and decreasing trend between *sp1_N* and *sp1_L* as well as between *sp1_L* and *sp1_E*.

IV. Automatic Model Construction

The model is usually too large to be hand-built. It is especially true for the temporal constraints that should comply with numerical equations of the domain. We propose an efficient algorithm that automatically generates the network of timed automata according to a simple system description requiring the number of components, their interactions and the number of qualitative levels (and their relative values). The algorithm is composed of two mains parts: the generation of the timed automata network and its simplification to make it more tractable.

A. Model Generation

We propose a qualitative model of the ecosystem part (species components) from a discretization of classical population dynamics equations. Components having only controllable events (fishing pressures) can be derived from any simple description since the change between qualitative levels is instantaneous. Figure 3 presents the automatic generation of the network of timed automata describing an ecosystem. Each timed automaton is built independently. The algorithm

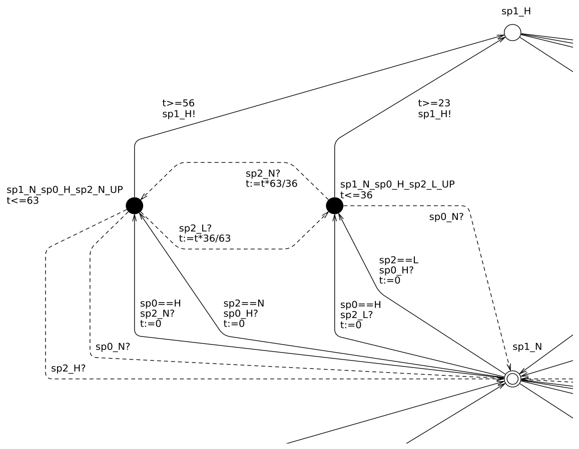

Fig. 2. Part of *SP1* species timed automata. Two stable locations (white circle) $sp1\_N$ (initial location) and $sp1\_H$ and two transient locations (black circles) denoting an increasing trends between the *Normal* and *High* levels. A clock t express the timing constraints of the evolution.

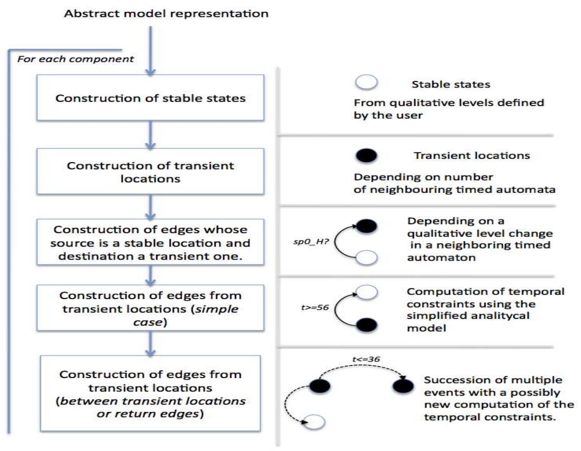
Abstract model representation
For each component
Construction of stable states
Construction of transient locations
Construction of edges whose source is a stable location and destination a transient one.
Construction of edges from transient locations (simple case)
Construction of edges from transient locations (between transient locations or return edges)
Stable states
From qualitative levels defined by the user
Transient locations
Depending on number of neighbouring timed automata
sp0_H?
Depending on a qualitative level change in a neighboring timed automaton
t>=56
Computation of temporal constraints using the simplified analitycal model
t<=36
Succession of multiple events with a possibly new computation of the temporal constraints.

Fig. 3. Automatic generation of timed automata algorithm

begins to build the stable locations and the transient locations depending on the possible evolutions in the neighbouring automata. Once the locations are defined, simple edges are created and finally return edges and edges between transient locations. The key point is to compute the guards on the edges and the invariants on the transient locations. According to the domain, these timed constraints could be difficult to obtain from experts. Given a discretized model and some simple parameters on the system, our idea is to use the analytical model to estimate the temporal information between two qualitative levels. The assumption that the current qualitative level of neighbours is constant as soon as their qualitative levels have not changed often allows a formal and efficient use of the original model.

In our ecosystem example, the timing constraints associated to transient locations are estimated using a simple parameterization routine based on the Lotka-Volterra equations that usually describe the population dynamics of biological systems [11]. We assume that the biomass level of the neighbouring species is constant as soon as they do not reach another biomass level. This assumption allows a simplification of the differential equations that enable its formal resolution. The three required parameters of these equations are well-known species information given by experts and sufficient for the computation of the timing constraints on species automata. A confidence interval is added on given parameters. Thus, given the state of the neighbouring automata, the simplified Lotka-Volterra model is used to evaluate the invariant of transient locations as well as the guard of the outgoing edges. It is also used to estimate the timed response of varying fishing pressures related to the exploited species, as fish pressure can be seen as a kind of predator. Finally, the same method is used to estimate the new clock values on edges between two transient locations. For the fishing pressure automata, timing constraints are deduced from fishing description as chronograms. For the environmental disturbances automata, temporal information is directly given by the user.

B. Model Simplification

After the generation step described above, the size of the automata network is huge. The exhaustive generation creates a lot of transient states between two stable states. The analysis of transient locations reveals a lot of redundancies in their temporal constraints (invariants and outgoing edges). The idea is then to reduce the size of the automata network by detecting similar transient locations and by merging them. A classical hierarchical clustering algorithm is applied to reduce the number of transient locations. This simplification is performed on ”similar” transient locations, which means that only locations having the same properties (qualitative properties, incoming and outgoing edges) but different invariants are merged. The model semantics is not modified but timing constraints are aggregated.

The distance used by this clustering algorithm is the following: $d = \sqrt{\Delta I^2 + \Delta G^2}$ such that ΔI (resp. ΔG) is the difference between invariants (resp. outgoing guards) of transient locations. Transient locations belonging to the same cluster are merged into one. The invariant of the resulting transient location is the invariant average of the locations belonging to the cluster. The outgoing edges of the cluster transient locations are also merged into one, having a guard as the average of the cluster guards. Since the entering edges do not support guards, their merging is not a problem. The maximum number of transient locations between two stable locations can be parametrized. Benchmarks on real-world ecosystems have shown that 12 transient locations between two stable locations is a good compromise to maintain the model accuracy while getting a decent time response.

V. QUERY PATTERNS

When dealing with qualitative models of large complex systems, the number of possible qualitative behaviors is huge as the size of the model exponentially grows with the number of components. To analyse or query the model, classical methods reach their limits, especially when explicit temporal constraints have to be managed. Model-checking is one of the most successful techniques for automatic verification of complex systems [14]. When model checking a timed automaton, the properties to be verified are usually expressed using the time logic TCTL (Timed Computation Tree Logic) defined by Alur and Dill [13]. TCTL formulae are defined using the following grammar:

$$f ::= p \mid x \in I \mid \neg p \mid p_1 \vee p_2 \mid \exists\Diamond_I\, p \mid \forall\Diamond_I\, p \mid \exists\Box_I\, p \mid \forall\Box_I\, p$$

where p is a property, $x \in \mathcal{X}$ is a clock and I is a time interval. The diamond operator $\Diamond p$ expresses that a path (i.e. a sequence of states) leads to a state satisfying the property p. The box operator $\Box p$ means that all the states on a path satisfy the property p. These modal operators can be combined with the universal quantifiers $\exists$ or $\forall$ over the paths.

WhichStates Pattern	Looks for all the possible situations at time t	for all S_i: $\exists\Diamond(S_i \wedge chrono.t = t)$
WhichDate Pattern	Looks for time t_i of the first occurrence of situation S	for all $t_i \in [0, t_{max}]$: $\exists\Diamond(S \wedge chrono.t = t_i)$
WhichDateSi Pattern	Variant of previous pattern, the initial situation is S and not S_{init}	if ($\exists\Diamond(S \wedge chrono.t = t)$) for all $t_i \in [t, t_{max}]$: $(S \wedge chrono.t = t) \Rightarrow \exists\Diamond(S' \wedge chrono.t = t_i)$
Safety Pattern	Looks if an undesirable situation S never happens	$not(\exists\Diamond S_i)$ or $\forall\Box(not S)$
Always Pattern	Determines whether a situation S is always satisfied	$\forall\Box(S)$
Stability Pattern	Look for a stable state S	$\forall\Box(S \Rightarrow \forall\Box S)$

TABLE I
SCENARIO PATTERNS AND THEIR TCTL EXPRESSIONS

Asking the stakeholders to directly query the model with TCTL properties is clearly too demanding. We identified six main query patterns that capture recurring queries a stakeholder would like to ask (cf. Table I). These queries can be expressed in TCTL and answered using model-checking algorithms.

VI. RESULTS

We present three kinds of results: benchmarks models, the developed Software called EcoMata and a brief description of real-case applications.
Some performance tests have been realized on the *Reachability* property on which rely the *WhichStates* and *WhichDate* patterns (the *Safety* and *Always* patterns take only few seconds whatever the model size). We have evaluated the response time of the UPPAAL model-checker (called VERIFYTA [15]) with

an increasing complexity of the global system model. Table II presents the time response (in seconds) of a reachability analysis. For each experiment we give the number of species and the number of fishing pressures of the ecosystem. We suppose that each species is described by four qualitative biomass levels and that each species interacts with three other network components (fishing pressures or other species). The number of locations and edges, as well as the number of locations of the biggest automaton of the network obtained after the automatic generation are given.

Species number	Fishing Press Number	Loc. number	Edges number	Max. Loc. automata number	Response Time in s *Reachability*
1	1	55	92	51	0.245
2	1	357	654	357	0.664
3	1	783	2146	717	2.090
4	2	836	2237	717	2.038
5	2	1184	2867	717	2.472
6	2	1644	4410	777	3.523
7	2	1902	4894	777	4.3
8	2	3484	9316	777	6.093

TABLE II
BENCHMARKS ON *Reachability* ANALYSIS

The large number of interactions between species and fishing pressures explains the size of the network (even after the reduction step). Up to 6 species, the model can be used in a very fluent way. An ecosystem having more than 8 species (with three interactions each) is hardly tractable in an interactive process.

EcoMata is the toolbox we developed in order to allow any stakeholder to design and explore his own prey-predator ecosystem. EcoMata, which is composed of three main parts: the network editor, the automata network generator and the query launcher, interacts with the UPPAAL model-checker [15] for the verification task. EcoMata is a free software[1] platform-independent written in JAVA.

This approach has been applied with success on a subsistence fishery in the Uvea coral reef lagoon in New-Caledonia [16]. A timed automata network has been developed to investigate the direct and indirect effects of various fishing strategies on the trophic network that contains five major lagoon species and three fishing forces. EcoMata has also been applied on a marine ecosystem in North Sea where a sequence of data was available for seven species (cod, haddock, herring, norway pout, saithe, sandeel, whiting) over 20 years. The idea was to create an ecosystem model initialized with the 1990's data and to compare the predictions given by EcoMata with the real data observed on sea. Given the uncertainties on such real data, the results reflects, for most of the species, the same trends on the dynamics.

VII. CONCLUSION

In this paper, we propose to model an ecosystem as a qualitative model represented by a network of timed automata. The qualitative model includes the integration of quantized continuous time subsystems and allows explicit timed constraints. The first contribution consists in building automatically the network of timed automata from a simple model description. One of the novelty is to give a way to control the complexity of the resulting model by detecting similarities in automata and by merging them using a hierarchical classification algorithm. The second contribution consists in a set of query patterns, expressed in the temporal logic TCTL and answering the most common requirements an user is asking when exploring a model. Experimental results demonstrate the effectiveness of model-checking algorithms for patterns based on reachability. Future work will focus on proactive scenarios using control synthesis techniques and the addition of new features such as costs to enriched the searched strategies.

[1]Ecomata can be downloaded from https://team.inria.fr/dream/ecomata/

REFERENCES

[1] J. Lunze, "Diagnosis of quantized systems based on a timed discrete-event model," *Trans. Sys. Man Cyber. Part A*, vol. 30, no. 3, pp. 322–335, May 2000.

[2] J. Dambacher, H. Li, and P. Rossignol, "Relevance of community structure in assessing indeterminacy of ecological predictions," *Ecology*, vol. 83, no. 5, pp. 1372–1385, 2002.

[3] P. Salles, B. Bredeweg, and S. Araújo, "Qualitative models about stream ecosystem recovery: Exploratory studies," *Ecological Modelling*, vol. 194, no. 1-3, pp. 80–89, 2006.

[4] B. Kuipers, *Qualitative reasoning: modeling and simulation with incomplete knowledge*. MIT Press Cambridge, MA, USA, 1994.

[5] L. Travé-Massuyès, L. Ironi, and P. Dague, "Mathematical foundations of qualitative reasoning," *AI Magazine, Special Issue on Qualitative Reasoning*, vol. 24, no. 4, pp. 91–106, 2003.

[6] C. Largouët and M.-O. Cordier, "Improving the landcover classification using domain knowledge," *AI Communication*, vol. 14, pp. 35–43, 2001.

[7] M.-O. Cordier and C. Largouët, "Using model-checking techniques for diagnosing discrete-event systems," in *Proceedings of the Twelfth International Workshop on Principles of diagnosis (DX'01)*, 2001, pp. 39–46.

[8] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, R. Mateescu, M. Page, and D. Schneider, "Analysis and verification of qualitative models of genetic regulatory networks: A model-checking approach," in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005, pp. 370–375.

[9] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *Proceedings of the 21th International Conference of Software Engineering*, 1999, pp. 411–420.

[10] P. T. Monteiro, D. R. Mateescu, A. T. Freitas, and H. de Jong, "Temporal logic patterns for querying qualitative models of genetic regulatory networks," in *European Conference in Artificial Intelligence (ECAI'08)*, 2008, pp. 229–233.

[11] J. D. Murray, *Mathematical Biology II: Spatial Models and Biomedical Applications*, ser. Interdisciplinary Applied Mathematics. Springer New York, 2003, vol. 18.

[12] R. Alur and D. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[13] R. Alur, C. Courcoubetis, and D. L. Dill, "Model-checking in dense real-time," *Information and Computation*, vol. 104, no. 1, pp. 2–34, 1993.

[14] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Progress on the state explosion problem in model checking," *Lecture Notes in Computer Science*, vol. 2000/2001, pp. 176–194, 2001.

[15] K. Larsen, P. Pettersson, and W. Yi, "Uppaal in a nutshell," *Journal of Software Tools for Technology Transfer*, vol. 1, no. 1-2, pp. 134–152, 1997.

[16] C. Largouët, M.-O. Cordier, Y.-M. Bozec, Y. Zhao, and G. Fontenelle, "Use of timed automata and model-checking to explore scenarios on ecosystem model," *Environmental Modelling and Software*, vol. 30, pp. 123–138, 2011.