Murdoch
UNIVERSITY

MURDOCH RESEARCH REPOSITORY
http://researchrepository.murdoch.edu.au/

Man, K. L., Krilavicius, T., Wang, K., Hughes, D. and Lee, K. (2011) *Modeling and Analysis of Radiation Therapy System with Respiratory Compensation Using Uppaal.* In: Proceedings of the 9th IEEE International Symposium on Parallel and Distributed Processing with Applications (IPSA 2011), 26 - 28 May, Busan, South Korea, pp 50 – 54.

http://researchrepository.murdoch.edu.au/4848/

Copyright © 2011 IEEE
Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Modeling and Analysis of Radiation Therapy System with Respiratory Compensation using Uppaal

Ka Lok Man
Xi'an Jiaotong-Liverpool University (XJTLU)
Dept. of Computer Science and Software Engineering
111 Ren'ai Road, Suzhou , Jiangsu 215123
ka.man@xjtlu.edu.cn

Tomas Krilavičius
Baltic Institute of Advanced Technologies
Saulėtekio 15, LT-10224, Vilnius, Lithuania
and Vytautas Magnus University
Informatics faculty
Vileikos 8, LT-44404, Kaunas, Lithuania
t.krilavicius@gmail.com

Kaiyu Wan and Danny Hughes
Xi'an Jiaotong-Liverpool University (XJTLU)
Dept. of Computer Science and Software Engineering
111 Ren'ai Road, Suzhou , Jiangsu 215123
{kaiyu.wan, daniel.hughes}@xjtlu.edu.cn

Kevin Lee
School of Information Technology
Murdoch University
Western Australia, 6150
kevin.lee@murdoch.edu.au

***Abstract*—The goal of radiation therapy is to give as much dose as possible to the exact target location and minimizing any dose to a normal tissue. Advances of Cyber-physical control systems allow planning and provide very accurate treatments. However, the current technology does not sufficiently compensate a respiratory movement, that is especially important in case of lung (area) cancer. In this paper we present a model of radiation treatment system developed to analyze a system that compensates respiratory motion. We use Uppaal, an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays, etc.).**

I. Introduction

The goal of radiation therapy is to give as much dose as possible to the exact target location and minimizing any dose to a normal tissue. Advances of Cyber-physical control systems allow planning and provide very accurate treatments. However, the current technology does not sufficiently compensate a respiratory movement, that is especially important in case of lung (area) cancer. Different techniques to cope with such problem are analyzed in [14]. Usage of gating combined with external surrogates is given an overview in [4]. However, most of the research models try to predict movement of the tumor are, e.g. [13], [21], [23]. This paper focuses on modeling and software, that is supposed to conform to the requirements, i.e. process images and move precisely and fast. We use formal methods for such analysis because they provide means for rigorous modeling and analysis of diverse systems. The main reasons for the population of the formal methods are as following.

- Unambiguous models Formal modeling languages allow the defining of systems unambiguously, because syntax and semantics are defined formally, and that includes means to define non deterministic and stochastic behavior precisely. Moreover, for the same reasons, unambiguous refinement and code generation techniques can be applied.
- Strict analysis techniques Because models are defined using languages with strict semantics, rigorous reasoning about models is possible. E.g., model checking, theorem proving and specifically designed algorithms can be used.

Various techniques and tools have been defined, e.g. process algebras [6], [11], [15], [16], [19], [22], timed automaton [2], hybrid automaton [1], SPIN [12] and Uppaal [3] tools and a lot more, see [5] for a wider overview. Successful application of formal techniques have been reported in different areas, such as the automotive industry [10], electronics [18], industrial devices control [17] and other.

In this paper we investigate the applicability of using timed automaton [2] and the Uppaal tool [3] for the design and functional analysis of a a radiation therapy system consisting of a HexaPOD couch with 6-degrees movement, a tracking camera, a marker (markers) and controller. Uppaal is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types and other convenient constructions. We specify a simplified model of the system, and analyze its functional properties, i.e. absence of deadlocks, liveness and safety.

In Section 2 we provide a detailed description of the radiation treatment system. Then we concisely introduce Uppaal and timed automaton in Section 3. In Section 4 we present and discuss a Uppaal model of the radiation treatment system and check some of its properties, and its applicability to further analysis. Finally Section 5 presents some conclusions and future work.

II. Radiation Treatment System

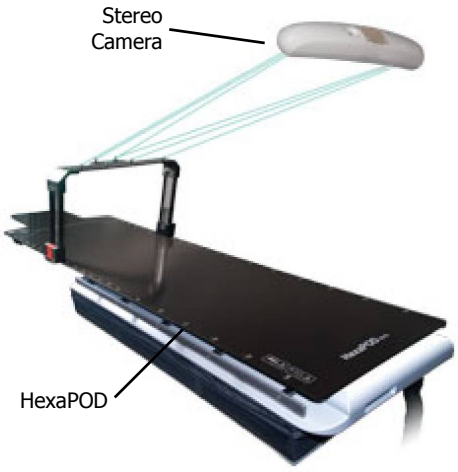

Figure 1. Radiation Treatment System.

Radiation treatment system under analysis[1] consists of the following components:

- Patient Setup Couch is used to position the patient for the treatment. In this study use HexaPOD couch [7], [9].
- External Radiation Beam Source is usually produced by a *medical linear accelerator*, in short, *linac*. In the current stage of our study it is not important, because we analyze controllability of the couch and tracking camera.
- Tracking Device provides information about the position of the patient. Different means and techniques can be used to perform it, see [14] for the details. In this paper we model a system with a stereo camera.
- Controller is a system, that controls all the process, in our case the controller uses information provided by the treatment plan and tracking device to control the HexaPOD.

Patient Setup Couch and Tracking Camera are depicted in Fig. 1.

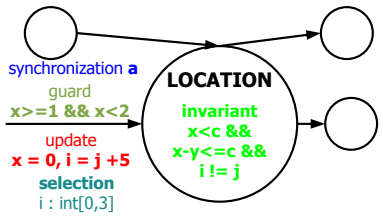

Figure 2. Timed Automata.

III. Timed Automata and Uppaal

Timed automata [2] is one of the most popular techniques for modeling and analysis of the real-time systems. We present a flavor of automata used in Uppaal [3]. Uppaal is an integrated tool environment for the modeling, simulation and verification of (complex) real-time systems. It is well-suited for systems that can be modeled as a collection of non-deterministic processes with finite control structure and real-valued clocks, communicating through channels or shared variables.

Definition 1. Let $\mathcal{C} = \{x, y, z, \ldots\}$ be a *set of clocks* and $\mathcal{B}(\mathcal{C}$ is the *set of clock restrictions* of the form $g, g_1, g_2 ::= x \bowtie c | x - y \bowtie c | g_1 \wedge g_2$ with $x, y \in \mathcal{C}, c \in \mathbf{N}$ and $\bowtie \in \{\leq, <, =, >, \geq\}$.

Definition 2. We will call a *timed automata* a finite directed graph $A = (L, l_0, \mathcal{A}, E, I)$ over $\mathcal{C}$ and $\mathcal{B}(\mathcal{C})$ where

- L is a *finite set of locations*;
- $l_0 \in L$ is the *initial location*;
- $\mathcal{A}$ is a *finite set of action names*;
- $E \subseteq L \times \mathcal{B}(\mathcal{C}) \times \mathcal{A} \times 2^{\mathcal{C}} \times L$ is a *finite set of edges*, and $I : L \to \mathcal{B}(\mathcal{C})$ assigns invariants to locations.

We will write $l \xrightarrow{g,a,r} l'$ instead of $(l, g, a, r, l) \in E$. l is called the *source location* of the state, g is the *guard*, a is the *action*, r is the *set of clocks to be reset* and l' is the *target location*. Timed automata can be represented as in fig. 2. Locations are drawn as nodes in the graph, and the initial location is usually marked with a double circle.

Definition 3. Let $A = (L, l_0, \mathcal{A}, E, I)$ be a *timed automata* over a set of clocks $\mathcal{C}$. We define the *timed transition system* $T(A)$ generated by A as $T(A) = (S, Act, \xrightarrow{tr})$, where:

- $S = L(\mathcal{C} \to \mathbb{R}_{\geq 0})$ is a *set of states* (l, v), where l is a location of the timed automata and v is a *clock valuation* that satisfies the invariant of l;
- $Act = \mathcal{A} \cup \mathbb{R}_{\geq 0}$ is the *set of labels*;
- two types of transitions are defined:
  - *action transitions* $(l, v) \xrightarrow{a} (l', v')$ such that exists an edge $(l \xrightarrow{g,a,r} l') \in E$ where v satisfies g, v' satisfies $v[r]$ and v' satisfies $I(l')$;

[1]There is a diversity of radiation treatment systems, see [14] for overview of the systems relevant to this study. However here we define just a selected setup.

– *delay transitions* $(l, v) \xrightarrow{d} (l', v')$ if $\forall d' \in [0, d] \Rightarrow v + d$ satisfies $I(l)$.

Let v_0 denotes the valuation such that $v_0(x) = 0, \forall x \in \mathcal{C}$. If v_0 satisfies the invariant of the initial location l_0, we shall call (l_0, v_0) the *initial state* of $T(A)$.

Timed automata are composed into a *network of timed automata*, consisting of n timed automata $A_i = (L_i, l_i^0, \mathcal{A}, E_i, I_i), i = 1...n$ over a set of clocks $\mathcal{C}$. Let $l = (l_1, ..., l_n)$ be a location of the network, then invariants are composed using conjunction $I(\bar{l}) = \bigwedge_{i=0}^{n} I_i(l_i)$.

Definition 4. Let $A_i = (L_i, l_i^0, \mathcal{A}, C, E_i, I_i), i = 1...n$ be a *network of* n *timed automata*. Let $\bar{l}_0 = (l_1^0, \ldots, l_n^0)$ be the *initial location vector*. Then the semantics is defined as a *transition system* $(S, s_0, \rightarrow)$, where $S = (L_1 \times \ldots \times L_n) \times \mathbf{R}^{\mathcal{C}}$ is the *set of states*, $s_0 = ((\bar{l}_0), v_0)$ is the *initial state* and transition relation contains three types of transitions:

- *time flow transitions* $(\bar{l}, v) \xrightarrow{d} (\bar{l}, v + d)$, if $\forall d' \in [0, d]$ holds $v + d' \models Inv(\bar{l})$;
- *discrete transitions :*
  - *synchronized* $((l_1, \ldots, l_i, \ldots, l_j, \ldots, l_n), v) \xrightarrow{\tau} \left((l_1, \ldots, l_i', \ldots, l_j', \ldots, l_n), v'\right)$ if $\exists i \neq j$, $\exists \left(l_i \xrightarrow{a!, g_i, r_i} l_i'\right) \in E_i$, $\exists \left(l_j \xrightarrow{a?, g_j, r_j} l_j'\right) \in E_j$, $v \models g_i \wedge g_j$, $v' \models v[r_i \cup r_j]$ and $v' \models I_i(l_i') \wedge I_j(l_j') \wedge \bigwedge_{k \neq i,j} I_k(l_k)$;
    The synchronization mechanism in UPPAAL is a hand-shaking synchronization : two processes take a transition at the same time. An edge with synchronization label $a!$ emits a broadcast on the channel a and that any enabled edge with synchronization label $a?$ will synchronize with the emitting process.
  - *asynchronous* $((l_1, \ldots, l_i, \ldots, l_n), v) \xrightarrow{\tau} ((l_1, l_i', \ldots, l_n), v')$ if $\exists \left(l_i \xrightarrow{a!, g_i, r_i} l_i'\right) \in E_i$, $v \models g_i$, $v' \models v[r_i]$ and $v' \models I_i(l_i') \wedge \bigwedge_{k \neq i} I_k(l_k)$.

There are two parts of syntax of Uppaal as shown below.

- One is the model syntax, which is in form of timed automata, which has been introduced briefly above. The modeling language offers additional features such as generation of *bounded integer variables* and *urgency*. The generation of bounded integer variables is shown as "select property" in edges of timed automata, and urgency means urgent locations and committed locations.
- The other kind of syntax in Uppaal is the declarations of projects and systems in Uppaal behind models of timed automata. These declarations are in form of expressions. Most syntax of the expression in declarations coincides with that of C, C++ and Java, but there are still a number of differences. For example, only unsigned integer number is valid in Uppaal, type of a variable can be defined as "char" or "string". Uppaal does not allow dynamic memory allocations, hence the pointers are not supported and the size of an array should be known at compile time, so symbol "*" is invalid in Uppaal. Another point to note is that, the formal parameters in function should not be variable-length array, which is not supported by Uppaal's syntax.

The query language of Uppaal, used to specify properties to be checked, is a subset of CTL (computation tree logic) [8]:

- `A[] property` *invariant*, property always holds in all paths;
- `A<> property` *eventually*, property holds in all paths at some moment;
- `<> property` *possibly*, property eventually holds at some state, at least in one path;
- `E[] property` *potentially always*, property eventually holds from some state, at least in one path;
- `p -> q` *leads to*, whenever p holds eventually q will hold;
- `deadlock` true, if *deadlock* state is reachable;
- `P.state` certain properties hold in the selected state.

IV. Uppaal Model of The Radiation Treatment System

We model a simplified version of the radiation treatment system defined in Section 2 that allows analyzing impact of latency as well as reaction of the HexaPOD to control inputs. Uppaal model consists of the following components: **Controller** that, based on the input from the tracking system, i.e. stereo camera, and its state, controls movement of the HexaPOD; **HexaPOD** that, moves according to its physical limitations and following the commands sent from the controller and **HexaPOD Buffer** that models asynchronous communication and latency between the controller and the HexaPOD.

A. HexaPOD

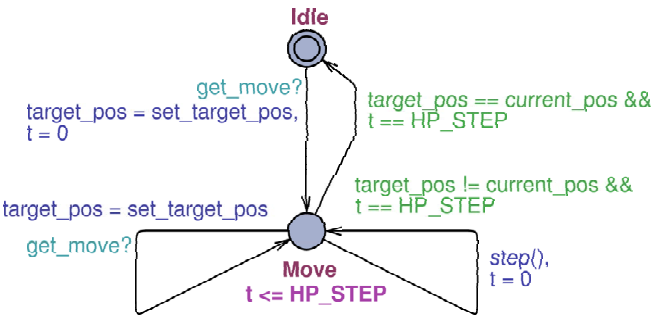

Figure 3. HexaPOD in Uppaal.

A simple model of HexaPOD is depicted in fig. 3. We model it as a one point-device, and only a discrete movement in x, y and z directions, and abstract from acceleration,

speed and rotation. Such simplified model still allows us to investigate impact of latency to simple trajectories, and general design of HexaPOD control. The model consists of two locations:

- Idle - HexaPOD waits for a command `move_to`. With this command it receives target location, and changes to **Move** location.
- Move - HexaPOD stepwise moves towards target, taking steps of predefined direction and of predefined distance. After each step it checks for a new target, and updates it. When the target is reached, it changes to **Idle** location.

B. HexaPODBuffer

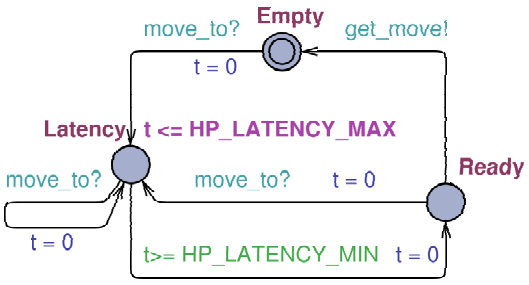

Figure 4. HexaPODBuffer in Uppaal.

HexaPODBuffer models asynchronous communication and latency. It consists of three location.

- Empty location denotes empty buffer, it awaits for an input from controller, i.e. `move_to` command, and the target, and then changes to **Latency**.
- Latency location is used to model delays in the system, i.e. after receiving new target the buffer delays for a while before providing it to the HexaPOD. However, the new target can be provided to buffer anytime.
- Ready location: when the buffer is ready, the target can be acquired by the HexaPOD using `get_move` command (action), and location is changed to **Empty**.

C. Controller

In this stage of analysis the controller just provides control commands to HexaPOD. It consists only of three locations: **Start**, **Move** and **Finished**, where the first and the third denote the beginning and the end of control, correspondingly. While in the **Move** location, the system sends control inputs stepwise provided by an array. Current version of the controller does not use timing.

D. Simulation and Analysis

Stepwise simulations allow us to gain a lot of insight into the model. However, Uppaal allows more, i.e. the conformance of the system to the selected properties can be verified. With this model we use the following properties:

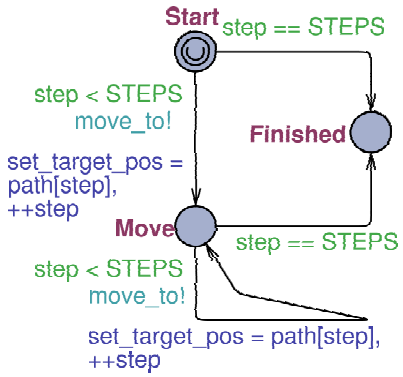

Figure 5. Controller in Uppaal.

- `E<> Controller.Finished` property allows to check, if there exists a path that allows for the Controller to reach its final location.
- `A<> Controller.Finished` allows to check, if the Controller reaches **Finished** location in all evolutions.
- `A<> Controller.Finished and Controller.step == Controller.STEPS` (or `E<> Controller.Finished and Controller.step != Controller.STEPS`) checks, if all control steps were performed before reaching the final state of the Controller.
- `E<> HexaPOD.Idle and HexaPOD.current_pos == Controller.path[Controller.STEPS-1]` checks, if HexaPOD reaches the target destination.

The last property can be modified to check, if in all cases the target is reached, but then an additional location can be added, as **Finished** in the Controller.

Provided properties allow checking different characteristics of the systems and producing diverse diagnostic traces. The traces can be compared to the required trajectories, and control properties of the HexaPOD as well as the Controller, estimated.

V. Conclusions and Future Plans

In this paper we have introduced the model of the radiation treatment system in Uppaal. It is a simplified model, including only some elements of the complete systems. Nevertheless, already some useful characteristics of the systems can be obtained. Moreover, it shows certain limitations of the approach, e.g. latencies in the order of of 30-40ms speeds of 16mm/s order require too different time scales, or accuracy becomes too low.

Our future plans are as follows. First we will make extensions of the Uppaal model for the case study. For example, model of HexaPOD should consider rotation, acceleration and velocity. Next we will add targeting component and implement the different control approaches. Furthermore, we

will provide continuous model of the HexaPOD to express its behavior more accurately. For example, we aim to either build more exact discrete model or generate discrete paths for the timed model. To model the same case study and seek the best modeling approach for it, we may consider using Behavioural Hybrid Process Calculus [15] or semi-formal control model in OpenModelica [20] for the Hybrid model. Finally we will combine the real respiratory movement trajectories and (formal) model to investigate systems adequacy to compensate it.

REFERENCES

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. *The algorithmic analysis of hybrid systems*, TCS, 138(1):3–34, 1995.

[2] R. Alur and D. Dill.*The theory of timed automata*, pages 45–73.

[3] G. Behrmann, A. David, and K. G. Larsen.*A tutorial on* UPPAAL, In M. Bernardo and F. Corradini, editors, Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004, number 3185 in LNCS, pages 200–236, sep 2004.

[4] R. I. Berbeco, S. Nishioka, H. Shirato, G. T. Y. Chen, and S. B. Jiang. *Residual motion of lung tumours in gated radiotherapy with external respiratory surrogates*, Physics in Medicine and Biology, 50(16):3655, 2005.

[5] J. P. Bowen and M. G. Hinchey, *Ten commandments of formal methods*, IEEE COMPUTER, 28:56–63, 1994.

[6] E. Brinksma, T. Krilavičius, and Y. S. Usenko, *Process Algebraic Approach to Hybrid Systems*, In Proc. of 16th IFAC World Congress pages 1–6, Prague, July 2005.

[7] H. Chung, H. Jin, T. Suh, J. Palta, and S. Kim. *Characterization of a commercial add-on couch, HexaPOD™ 6D robotic treatment couchTOP*, In R. Magjarevic, R. Magjarevic, and J. H. Nagel, editors, World Congress on Medical Physics and Biomedical Engineering 2006, volume 14 of IFMBE Proceedings, pages 1945–1947. Springer Berlin Heidelberg, 2007, 10.1007/978-3-540-36841-0_485.

[8] E. Clarke, O. Grumber, and D. Peled. *Model Checking*, MIT, 2001.

[9] Elekta. *Elekta Synergy® S with HexaPOD™*, On-line, 2006.

[10] B. Gebremichael, T. T. Krilavičius, and Y. S. Usenko. *A formal model of a car periphery supervision system in* UPPAAL. In Proc. of Workshop on Discrete Event Systems (WODES'04), pages 433–438, Reims, France, sep 2004.

[11] C. A. R. Hoare. *Communicating Sequential Processes*. Prent.-Hall, 1985.

[12] G. J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley, sep 2003.

[13] A. Kalet, G. Sandison, H. Wu, and R. Schmitz. *A state-based probabilistic model for tumor respiratory motion prediction*. Physics in Medicine and Biology, 55(24):7615, 2010.

[14] P. J. Keall, G. S. Mageras, J. M. Balter, R. S. Emery, K. M. Forster, S. B. Jiang, J. M. Kapatoes, D. A. Low, M. J. Murphy, B. R. Murray, C. R. Ramsey, M. B. V. Herk, S. S. Vedam, J. W. Wong, and E. Yorke. *The management of respiratory motion in radiation oncology report of aapm task group 76*, Medical Physics, 33(10):3874–3900, 2006.

[15] T. Krilavičius. *Hybrid techniques for hybrid systems*, PhD thesis, Enschede, 2006.

[16] T. Krilavičius and K. Man. *Intelligent Automation and Computer Engineering*, chapter Behavioural Hybrid Process Calculus for Modelling and Analysis of Hybrid and Electronic Systems. Springer, 2009.

[17] T. Krilavičius and V. Miliukas. *Functional modelling and analysis of a distributed truck lifting system*, In The 5th International Conference on Electrical and Control Technologies (ECT 2010), page 6, Kaunas, Lithuania, 2010.

[18] K. Man, T. Krilavičius, C. Chen, and H. Leung. *Application of bhave toolset for systems control and mixed-signal design*. In Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS), Hongkong, March 2010.

[19] R. Milner. *Communication and Concurrency*. Pren.-Hall, 1989.

[20] OpenModelica, url = http://www.openmodelica.org, year = 2011, howpublished = Web pages,.

[21] D. Ruan, J. A. Fessler, and J. M. Balter. *Real-time prediction of respiratory motion based on local regression methods*, Physics in Medicine and Biology, 52(23):7137, 2007.

[22] D. A. van Beek, K. L. Man, M. A. Reniers, J. E. Rooda, and R. R. H. Schiffelers. *Syntax and Consistent Equation Semantics of Hybrid Chi*. JLAP, 68(1-2):129–210, 2006.

[23] H. Wu, G. C. Sharp, B. Salzberg, D. Kaeli, H. Shirato, and S. B. Jiang. *A finite state model for respiratory motion analysis in image guided radiation therapy*, Physics in Medicine and Biology, 49(23):5357, 2004.