IJCSI International Journal of Computer Science Issues, Volume 13, Issue 3, May 2016
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org http://dx.doi.org/10.20943/01201603.4553

Surgical Robots: Behaviors Specification and reasoning

Ammar Mohammed[1], Luzie Schreiter[2]

[1]Deparment of Computer Science, Institute of Statistical Studies and Research, Cairo University, Egypt
Department of Computer Science, Arab East Colleges, Riyadh, KSA

[2] Institute for Process Control and Robotic, Karlsruhe Institute of Technology,
76131 Karlsruhe, Germany

Abstract

A Surgical robotic system is a computer assisted electro-mechanical device in a surgical context which helps the surgeon by performing interventions. This system is a safety critical system where occurring of errors might lead to the endangerment of human life, or substantial economic loss. Therefore, methods that guarantee the correctness of the system adherence to crucial safety properties are needed. A quite new way that can be used to specify the dynamics behaviors of surgical robots is to use hybrid automata. The semantics of hybrid automata allows us to reason about and to simulate the behavior of robots. This paper shows how to use hybrid automata to model a case study on the coordination between surgical robots working in a shareable area. Additionally the paper shows some formal analysis of particular requirements on both simulation and model checking level to reason about the behavior of the robots.

Keywords: Surgical robots, formal specification, hybrid automata

1. Introduction

Surgical robotics system is an area which received attention in the area of robotics. A robotic Surgery is defined as a computer assisted electro-mechanical device in a surgical environment which supports the surgeon by performing interventions [12]. One of the most famous surgical robotic systems is the da Vinci system which is used for minimally invasive robotic Surgery [8]. The da Vinci consists of up to three robotic arms for holding surgical equipments, a patient side cart including one robotic arm equipped with an endoscopic camera. The main purpose of the systems is to perform the surgical intervention.

Generally, Surgical robotic systems are consider safety critical systems where occurring of errors might lead to the risk of human life, to economic loss, or to a huge environment danger [13]. Therefore, specifying the behavior of those systems has to be carried out carefully in order to avoid the side effects that might cause undesirable or even fateful behaviors. One of the primary concerns is the safety of the patient, which in turn leads to an extreme reliable, safe and robust robotic system. To our knowledge, no safety standards for robotic surgical systems are established yet. There is no common definition of the term safety in the context of robotic surgery and it is not easy to defined. In our understanding, the correctness, the reliability and the dependability are quality indicators for the safety of surgery robotics systems.

On the other hand, formal methods based on mathematical models are very convenient to design safety critical systems. They have been used extensively in different safety critical applications [9, 21]. The main advantage of formal methods is that they provide a precise and obvious description of the behavior of the system under design. Additionally, they not only allow us to formally specify systems at different levels of abstraction, but also to verify the consistency of the specified systems before involving in the implementation phase. Usually Formal methods use a kind of state transition diagrams, e.g. finite automata or state-charts, to specify the discrete behaviors of system under consideration. One notable power of such transition diagrams is that they grant themselves to

formal analysis by means of the so called model checking [6]. However, in the context of surgical robotics, where the robots are situated in realistic physical environment, it is necessary to consider continuous behaviors in addition to discrete behaviors. An examples of those continuous behaviors includes the movement of the robot, the kinetic movement of the arms. Unfortunately, the traditional state transition diagrams are not sufficient to combine these types of behaviors. *Hybrid automata* [10], in contrast to state transition diagrams, offer an elegant methods to capture such types of behaviors.

Broadly speaking, hybrid automata extend the regular state transition diagrams with differential equations to handle those continuous actions. On one hand, the state transition diagrams are used to represent the discrete changes of behaviors, while the differential equations are used to model the continuous changes. The formal semantics of hybrid automata make them accessible to formal verification using model checking. Accordingly,several model checkers have been implemented to reason about the hybrid automata models [11, 7, 2, 17].

The contribution of this paper is to approach hybrid automata to surgical robots context by modeling a particular scenario between two surgical robots in a shared environment. The paper is doing so by modeling the mutual exclusion scenario between two robots sharing a critical working area. In particular, the paper focuses on the application of the mutual exclusion protocol at the OP:Sense Setup [20]. The paper additionally shows some formal analysis of particular requirements on both simulation and model checking level.

The paper is structured as follows: Sec. 2. discusses some related work. Sec 3. gives a theoretical background of hybrid automata. In section 4. a surgical robots case study is defined and the model of the scenario is explained in details using hybrid automata. In Sec. 5. a formal analysis of the modeled scenario is described. Finally, Sec. 6. concludes the paper.

2. Related Work

Generally, in the areas of robots there are existing some standards. For example, there are standards that describe the safety requirements for industrial robots. Additionally, there are standards for the medical devices that describe requirements of the quality management system of medical devices. To our knowledge, however, no safety standards for robotic surgical systems are settled. The use of formal methods to specify critical safety in the area of surgical robots is quite new, although they are used in different safety critical areas such like air traffic control [9], vehicle control and software development [21]. Different papers attempted to use a certain kind of formal method to model surgical robots. For example, in [5] the authors approached "Hybrid Input/Output automata" for the verification of robotic surgical system where components invariant were used to prove an overall safety property. In [18] the authors presented the application of autonomous surgery and verification of the surgical plan for a simple puncturing. In [14] the authors proved a control algorithm for directional force feedback in a surgical context. Compared to the work of this paper, neither of these previous work models the continuous behaviors of the Surgical robots nor provides formal analysis using model checking. A recent work similar to our work is presented in [4], the author presented hybrid automata model on a very simple puncturing case study and performed formal analysis using reachability analysis. However, our case study has more interactions and coordinations between robots. Additionally, we propose to model more complex dynamics scenarios and use the powerful of simulation to observe the dynamic behaviors that are beyond the hybrid automata to verify.

3. Hybrid Automata

A hybrid automaton [10] is represented graphically as a state transition diagram dialect like state-charts, augmented with mathematical formalisms on both transitions and locations (see Fig. 2). Formally speaking, a hybrid automaton is defined as the following (more details about Hybrid automata with illustrative examples can be found in [10]).

Definition 1 *A hybrid automaton is a tuple* $H = (X, Q, Inv, Flow, E, Jump, Reset, Event, Init)$ *where:*

- $X \subseteq \mathbb{R}^n$ *is a finite set of n real-valued variables that model the continuous evolutions of the automaton with respect to time.*
- Q *is a finite set of locations.*

- *$Inv(q)$ is the invariant predicate, which assigns an invariant conditions for each location $q \in Q$. The control of a hybrid automaton stays inside the location q, as long as the invariant condition $Inv(q)$ satisfied.*

- *$Flow(q)$ is the flow predicate on variables X for each location $q \in Q$, which defines how the variables in X evolve over the time inside q. It constrains the time derivative of the continuous part of the variables at location q. Basically, we represent the flow as a constraint relation of the real variables to the time. In the graphical representation, a flow of a variable x is denoted as $\dot{x}$. A hybrid automaton is classified according to the kind of the flow into different classes including timed automata [1],linear automata, rectangular and non-linear automaton.*

- *$E \subseteq Q \times Q$ is the discrete transition relation over the locations. Each edge $e \in E$ is augmented by the following annotations:*

  Jump: *is a condition over X that must hold to enable transitions. Omitting a jump condition on a transition means that the jump condition is always true and it can be taken at any point of time.*

  Reset: *is a constraint, which may reset the variables to certain assignments. In the graphical representation, $X' = v$ denotes that the variable X is reset with the value v. Resetting variables are omitted on transition, if the values of the variables do not change before the control jumps from a location to another location.*

  Event: *synchronization label, used to synchronize and coordinate the behavior of several automata. These synchronization labels are used to construct the composition of the automata.*

- *Init is the initial condition that assigns initial values to the variables X to each control location $q \in Q$.*

Informally speaking, the semantics of a hybrid automaton is defined as a labeled transition system between states, where a state consists of the current location of the automaton and the current valuation of the real variables. To model behaviors of larger systems, each part of a system is modeled using hybrid automaton, and the communication between the parts is performed by means of shared variables and synchronization labels. The behavior of the overall system is controlled using the so-called parallel composition. Our scenario is composed of four automata (see Fig.1) as we will show in the next sections.

4. Computer and Robot Assisted Surgery Scenario

The setup of our scenario is a reactive surgical multi-agent systems consisting of four main components: two surgical robots operating in a shared area, controller and scheduler. The main goal of the system is to provide coordination between the two robots on the shared area and to avoid any collision that might occur between the robots. Additionally, the system provides a cooperation between the two robots and a controller in such a way that both robots read the environment and send the result to single processor on the controller. To achieve the previous purpose, a scheduler (a part of the controller) coordinates the behavior of both robots when they request sending the data to the controller.

In order to prevent the robots form collision on a critical shared area, a part of the two robots should model mutual exclusion protocol based on Fisher [15]. Also, each robot repeatedly reads its environment and sends the result to the controller. Both robots share a single processor with the controller. In our scenario, the scheduler coordinates the reading of both robots. In case of both robots require reading process at the same time, the scheduler secures that the second robots has priority over the first robot. That means the second robots can interrupt the first robot reading process. As soon as the second robot completes its reading, the first robot reschedules its reading process.

Generally, after sending its reading, each robot is delayed for a certain time to operate its environment and to mutually coordinates the shared working area before starting a new reading process. If the reading process is not sent to the controller in adequate time, the robot takes in consideration to send it again.

4..1 Modeling the Scenario

The previous scenario will be modeled using four hybrid automata (see Fig. 1). The automata communicate between each others through shared variables and synchronization events. Since the model include two robots, we use the index i to represent each the robots.

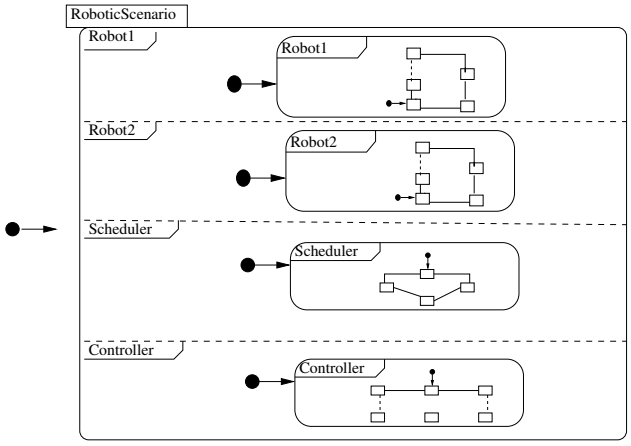

Figure 1: A surgical robotic scenario in abstract level.The shape ⬭—⬭ indicates the hidden sub states.

In what follow, we describe details of each automaton participating in the Fig. 1.

4..1.1 The robot behavior

The behavior of the two robots (see Fig.2) are identical in the the model except that each robot$_i$ uses a continuous variables y_i to measure the time constraints inside locations. Both robots coordinate their behavior using the shared variable k which mutually controls the robots within the shared area. The robot$_i$ has seven locations: **Init**, **move**$_1$, **move**$_2$, **SA**, **read**, **wait**, and **send**. Initially, the control of the robot$_i$ is in **Init** with y_i *:=0* and *k:=0*. In the location **move**$_1$, the robot has to wait no more than a constant time unit, indicated by the parameter a, to update the value of the variable k, i.e. assigning *k:=i*. In the location **move**$_2$, the robot waits a lower time bound delay, indicated by the parameter b, before accessing the shared location **SA**. After remaining in the later location for acceptable delay, the robot starts to read a request using the event *request*$_i$ and resets y_i and k to zero. In the location **read**, the robot waits until the event *read*$_i$ takes place from the scheduler in order to construct the reading. Once the reading is allowed, the control moves to the location **wait**. In the previous location, the robot remains inside for up to 4 time units. During this time, the robot is ready to send its reading to the controller using the event *send*$_i$. The previous event occurs as soon as the controller is ready to receive the reading. If the robot did not receive the event *send*$_i$ in the due time, it starts to initiate a new reading-request and goes back to location *read*. After sending the reading, the robot stays inside the location **send** waiting for acknowledgment from the controller using the event *ack*$_i$, and then it moves back to the location **Init** and resets the variable to y_i *:=0* and *k:=0*.

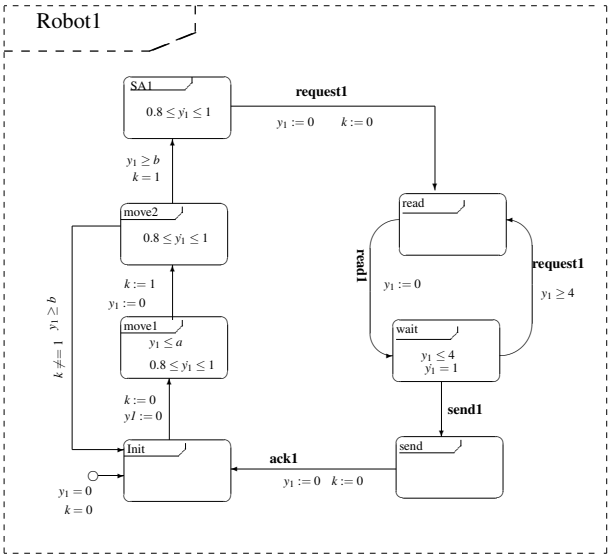

Figure 2: Hybrid Automaton - Robot

4..1.2 The scheduler

The Scheduler automaton (see Fig. 3) is responsible for allocating each robot CPU time on the shared processor. For this purpose, it uses the event *request*$_i$ to synchronize both robots. This event indicates that the robot$_i$ is initiating a request for CPU time to construct a reading. The scheduler automaton uses two variable x_1 and x_2 to

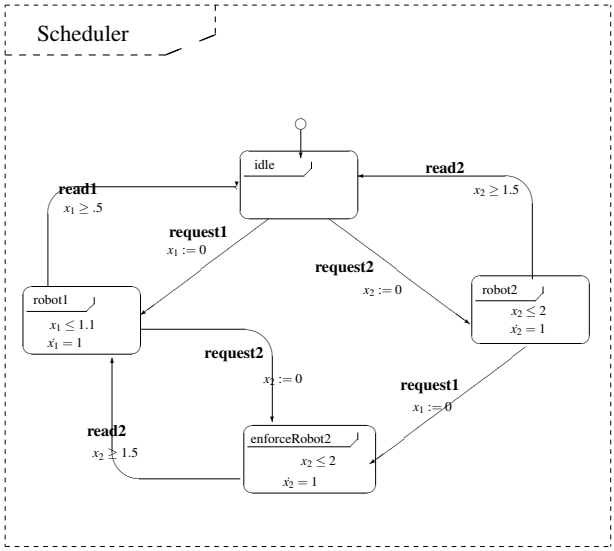

Figure 3: Hybrid Automaton - Scheduler

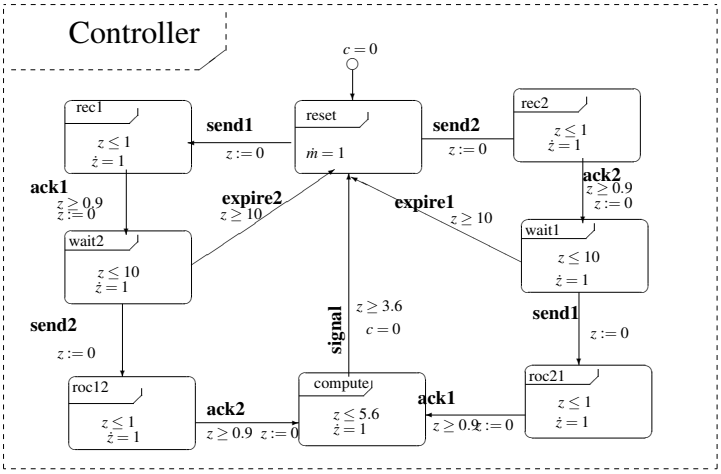

Figure 4: Hybrid Automaton - Controller

model the time that robot_1 and robot_2 have received since the last request. The first robot takes duration between 0.5 and 1.1 milliseconds for the process of reading the environment, while the second robot takes duration between 1.5 and 2.0 milliseconds. The scheduler has four locations namely: **idle**, $\textbf{robot}_1$, $\textbf{robot}_2$, and **enforceRobot2**. These locations responsible for the variations of requests. Initially, the control starts at location **idle** as long as there is no pending request. As soon as each of the robots requests a reading, the control goes to location either the location $\textbf{robot}_1$ or $\textbf{robot}_2$. Whereas the control goes to the location **enforcerRobot2**, when both robots have pending requests. In case of both robots request the CPU, a priority is given to the robot_2.

4..1.3 The Controller

The Controller automaton (see Fig. 4) uses the variable z to control its behavior. Initially, the control starts in the location **idle** waiting for the send-signal from each of the robots. Once the signal is acknowledged, the control goes to either the location $\textbf{wait}_1$ or $\textbf{wait}_2$ based on the robot which fire the signal. In the previous location, the automaton has to wait for send-signal up to 10 time units. If $signal_2$ event, respectively $signal_1$ does not take place in the required time, the invariant $z \leq 10$ forces the control to jump back to the location **idle** by firing the event $expire_1$ respectively $expire_2$. However, If the event send occurs in the due time, it is acknowledged and the control transfers to the location **compute**. In the later location, the controller automaton measures the time needed to calculate the command to send it to the appropriate robot. Additionally, the controller is augmented with a clock variable c which measures the time elapsed since the last signal has been sent to the robot or since the run has begun when there was no signal sent.

5. Formal Analysis

Having defined and modeled our proposed scenario using hybrid automata, one can analyze the behaviors of the model using either simulation or formal verification tools. For this purpose, there are a wealth of tools and languages proposed to analyze the behavior of systems modeled by means of hybrid automata. Simulation tools, for example [19], analyze intended behavior of under investigation using several runs of the model. Simulation is powerful to analyze more complex continuous dynamics.

Formal verification tools, on the other hand, are appealing as a concept to guarantee design correctness, but they are limited to less continuous dynamics. In this section we first investigate the formal analysis based on the simulation. A step further, we exploit the formal analysis of the scenario with the help of model checking; in particular using one of standard model checkers of hybrid automata HYTECH [11].

5..1 Simulation of The Scenario

We simulate the described scenario in Matlab Simulink State-charts. Therefore we used two lightweight robot models and modeled the critical structure as a sphere. Both robots are not allowed to enter the critical structure at the same time. Since the second robot is the prioritized ,the critical structure is entered only from this robot if both robots are requested to enter, see Fig. 5. The notation of state-charts allows us to precisely specify the state-based systems. Hence it is possible to show the fundamental behavior of the case study. But to prove other system requirements, especially time specification, it is necessary to use other verification techniques. Therefore we decided to use model checking tools, which are described in the following section.

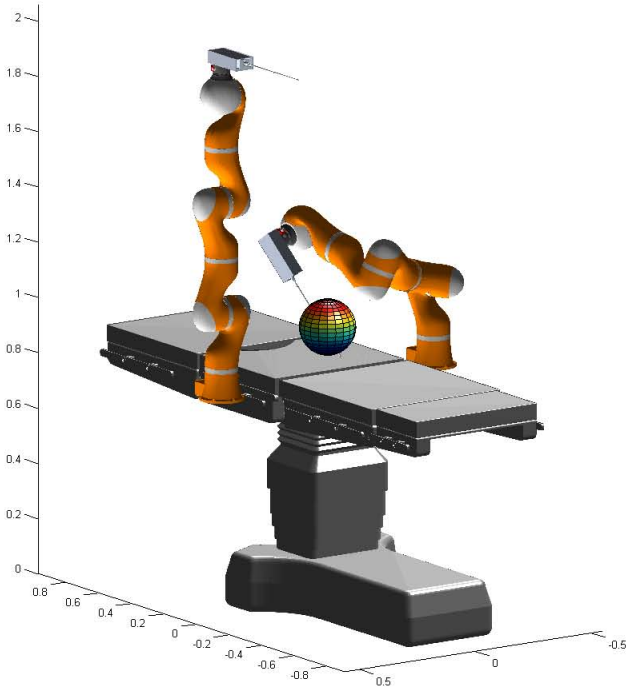

Figure 5: Simulation - Robot 2 is entering the critical structure

5..2 Model Checking

As we already mentioned, hybrid automata are equipped with formal semantics, which make them possible to applicable not only in simulation but also in formal methods. This allows us to prove certain properties of the specified systems, e.g. by model checking. However, in the context of hybrid automata the term *model checking* usually refers to *reachability* testing, i.e. the question whether some (undesirable) state is reached from the initial state of the model. A model checker first computes the reachability of the state space, and then checks the reachability of a certain property simply by intersecting the reachable state space and the property.

For the behavior specification of our scenario, we can conduct various experiments using the model checker HYTECH [11]. HYTECH is implemented for formal verification of linear hybrid automata. It takes, as input, textual representations of hybrid automata like the one showed in the Fig. 6 and performs reachability tests on the state space of the automata representing the overall model. All the kinds of properties specified in terms of reachability analysis can be checked using the model checker. A property of interest can be formally specified using extended versions of of temporal logics [3]. Following the specification formalism of the logic RCTL [16], a reachability of a certain property Ψ asserts that starting from an initial state, is there a state along a run of the model in which Ψ is satisfiable ?. This can be specified in RCTL as follows:

$$init \rightarrow \exists\Diamond\Psi$$

A safety property Ψ which states that *something bad must never occur* can be specified as in terms of reachability as the following formula:

$$init :\rightarrow \forall\Box\neg\Psi.$$

```
1  automaton Robot1
2    synclabs:  request1, read1, send1,ack1;
3    initially  Init & y1=0 & k=0;
4   loc Init:
5     while True wait {}
6     when k=0  do {y1' = 0} goto move1;
7   loc move1:
8      while y1<=a wait {dy in [8/10,1]}
9      when True do {k'=1, y1'=0} goto move2;
10 loc move2:
11     while True wait {dy in [8/10,1]}
12     when y1 >= b & k=1  goto SA1;
13    when y1 >= b & k=0  goto Init;
14    when y1 >= b & k=2  goto Init;
15 loc SA1:
16     while True wait {dy in [8/10,1]}
17 when True do {k'=0, y1'=0} sync request1 goto read;
18  loc read:
19    while True wait {}
20    when True  do {y1' = 0} sync read1 goto Wait;
21 loc Wait:
22     while y1=< 4 wait {dy =1}
23     when y1 >= 4  sync request1 goto read;
24    when True  sync send1 goto Send;
25 loc Send:
26     while True wait {}
27     when True sync ack1 goto Init;
28 end

29 init_reach := reach forward from init endreach;
30 ext_error := loc[Robot1] = SA1  & loc[Robot2] = SA2;
31   if not empty(init_reach & ext_error)
32    then prints "Mutual Exclusion is violated";
33 else prints "Mutual Exclusion holds";
34 endif;
```

Figure 6: Example from the HYTECH code of a surgical Robot (Lines 1–28) Fig. 2. Some analysis commands are shown in lines 29–34.

our main concerns in the safety property is to prove that the formula Ψ is never reached from the initial states.

The question: **is the Mutual exclusion guaranteed ?** is consider a safety property. Using HYTECH, it is easy to prove this type of property by checking that the two robots can not be found in the shared area at the same time, i.e., no reachable state satisfies the *robot1* resides in location *SA1* and *robot2* resides in location *SA2* at the same time. Figure 6 (line 29-33) shows how to check this property with HYTECH.

The model checkers can be used not only to check reachability of a certain property, but also to find the time requirements between specific events. For example, HYTECH can be used to determine the delay between signals that have been sent to the robots. In particular, we can use HYTECH to compute the value of the of the clock *c* in the set of all reachable states of the automaton *controller*.

HYTECH additionally can be used not only to check safety requirements, but also to find the constraints on specific parameters to guarantee the existence of safety property. The later is very a suitable technique for designing parameters of robots specially when they are situated in a critical environment. For example, HYTECH can compute the condition on the parameter **a** and **b** of the two robots to guarantee the existence of mutual exclusion.

6. Conclusion

The Area of surgical robots received recent interest as an area of autonomous robots. Designing surgical robots must take into consideration lots of safety concerns in order to avoid any disasters that might happen. An elegant method that help to design correct behaviors of systems and to guaranteed the safety is to use formal methods. This paper presented a formal way to model and specify a complex scenario of surgical robots by means of hybrid automata. In particular, the paper showed how to coordinate the behaviors of two robots operating in a shared critical area. Hybrid automata are used to model a scenario based on a mutual exclusion protocol. The paper simulated the described scenario with Matlab/Simulink. Since the surgical robots application requires procedures for guaranteeing the safety, the paper additionally used the rigorous model checking technique on hybrid automata to verify the correctness of the model.The later technique can be used to designing parameters of the robots in order to guarantee the safety correctness of the system. The successful formal analysis indicates that approaching hybrid automata to formal model more complex scenario in the area of surgical robots is desirable.

References

[1] R. Alur and D. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[2] G. Behrmann, A. David, and K. G. Larsen. A tutorial on Uppaal. In M. Bernardo and F. Corradini, editors, *Proceedings of 4th International School on*

Formal Methods for the Design of Computer, Communication, and Software Systems – Formal Methods for the Design of Real-Time Systems (SFM-RT), LNCS 3185, pages 200–236. Springer, Berlin, Heidelberg, New York, 2004.

[3] M. Ben-Ari, A. Pnueli, and Z. Manna. The temporal logic of branching time. *Acta Informatica*, 20(3):207–226, 1983.

[4] D. Bresolin, L. Geretti, R. Muradore, P. Fiorini, and T. Villa. Verification of robotic surgery tasks by reachability analysis: A comparison of tools. In *Proceedings of the 2014 17th Euromicro Conference on Digital System Design*, DSD '14, pages 659–662, Washington, DC, USA, 2014. IEEE Computer Society.

[5] M. Capiluppi, L. Schreiter, J. Raczkowsky, and H. Woern. Modeling and verification of a robotic surgical system using hybrid input/output automata. In *press European Control Conference, Zurich*, 2013.

[6] E. Clarke, O. Grumberg, and D. Peled. *Model checking*. Springer, 1999.

[7] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control, 8th International Workshop, Proceedings*, LNCS 3414, pages 258–273. Springer, Berlin, Heidelberg, New York, 2005.

[8] G. S. Guthart and J. K. Salisbury Jr. The intuitive¡ sup¿ tm¡/sup¿ telesurgery system: overview and application. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 618–621. IEEE, 2000.

[9] A. Hall. Using formal methods to develop an atc information system. In M. Hinchey and J. Bowen, editors, *Industrial-Strength Formal Methods in Practice*, Formal Approaches to Computing and Information Technology (FACIT), pages 207–229. Springer London, 1999.

[10] T. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292, New Brunswick, NJ, 1996. IEEE Computer Society Press.

[11] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. In *CAV '97: Proceedings of the 9th International Conference on Computer Aided Verification*, pages 460–463, London, UK, 1997. Springer-Verlag.

[12] D. Herron and M. Marohn. A consensus document on robotic surgery. *Surgical endoscopy*, 22(2):313–325, 2008.

[13] J. C. Knight. Safety critical systems: challenges and directions. In *ICSE 2002. Proceedings of the 24rd International Conference on Software Engineering 2002.*, pages 547–550. IEEE, 2002.

[14] Y. Kouskoulas, D. Renshaw, A. Platzer, and P. Kazanzides. Certifying the safe design of a virtual fixture control algorithm for a surgical robot. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, HSCC '13, pages 263–272, New York, NY, USA, 2013. ACM.

[15] L. Lamport. A fast mutual exclusion algorithm. *ACM Transactions on Computer Systems (TOCS)*, 5(1):1–11, 1987.

[16] A. Mohammed and U. Furbach. Mas: Qualitative and quantitative reasoning. In L. Dennis, O. Boissier, and R. Bordini, editors, *Programming Multi-Agent Systems*, volume 7217 of *Lecture Notes in Computer Science*, pages 114–132. Springer Berlin Heidelberg, 2012.

[17] A. Mohammed and C. Schwarz. Hieromate: A graphical tool for specification and verification of hierarchical hybrid automata. In M. H. B. Mertsching and Z. Aziz, editors, *KI 2009: Advances in Artificial Intelligence, Proceedings of the 32nd German Conference on Artificial Intelligence*, LNAI 5803, pages 695–702. Springer, 2009.

[18] R. Muradore, D. Bresolin, L. Geretti, P. Fiorini, and T. Villa. Robotic surgery: Formal verification

and plans. *Robotics Automation Magazine, IEEE*, 18(3):24 –32, sept. 2011.

[19] S. Neema. Analysis of matlab simulink and stateflow data model. Technical report, Tech. Rep. ISIS 01-204, Vanderbilt University, Nashville, TN, 2001.

[20] P. Nicolai, T. Brennecke, M. Kunze, L. Schreiter, T. Beyl, Y. Zhang, J. Mintenbeck, J. Raczkowsky, and H. Wörn. The op:sense surgical robotics platform: first feasibility studies and current research. In *International Journal of Computer Assisted Radiology and Surgery*, pages 136–137, 2013.

[21] A. Sobel and M. Clarkson. Formal methods application: an empirical tale of software development. *Software Engineering, IEEE Transactions on*, 28(3):308–320, 2002.