2013 European Control Conference (ECC)
July 17-19, 2013, Zürich, Switzerland.

Modeling and Verification of a Robotic Surgical System using Hybrid Input/Output Automata

Marta Capiluppi[2], Luzie Schreiter[1], Paolo Fiorini[2], Joerg Raczkowsky[1], Heinz Woern[1]

***Abstract*— The area of robotic surgical systems has to deal with several important safety aspects to ensure that the patient and the Operating Room staff are safe. A robotic surgical system has to fulfill specific safety requirements and to ensure that the system reacts like its specification. To this end, a verification process is necessary. In this paper an architecture for robotic surgery is modeled using the framework of Hybrid Input/Output Automata (HIOAs). A case study based on a surgical robotic operation scenario is presented and modeled using HIOAs. Exploiting the modularity and compositionality theory of HIOAs, the verification of the system is performed.**

***Index Terms*— Robotic Surgery, Hybrid systems, Hybrid I/O Automaton, Verification.**

I. INTRODUCTION

Surgical robots were first introduced for neurosurgical intervention guidance, and have later been employed for laparoscopy, prostatectomy, and orthopaedic surgery. More recently, applications of robotic technology to surgery were aimed at neurosurgical procedures. Finally, surgical robots for soft tissues and more general procedures were introduced into the market. The use of robots in surgery helps the surgeons in achieving basic and repetitive tasks, leaving to the experience of the medical staff more difficult and context-related operations.

Hence surgical robots need to cooperate with the medical staff (surgeons and nurses) to perform the surgical operation and interact with patients. This leads to the necessity of improving the safety aspects of the man-machine interaction. Indeed safety is an important aspect both of surgical and robotic systems [1]. Indeed, in many other related fields there are different approaches to safety verification, for instance in the area of autonomous vehicles [2] and in the area of domestic robots [3]. Safety verification includes the application of methods, procedures, tests and other evaluations, to determine whether a system is or has been operating as intended.

Our goal is to describe a procedure for verifying safety in a surgical robotic system that is able to deal with the issues created by a medical context. Safety is related to reliability of the system under study, hence to redundancy and component replacement. We start from an architectural design that should enable the verification of some component properties that are maintained in their composition, in order to prove that the system is still behaving in the desired way without compromising the safety property.

In particular, since a surgical robotic system features both continuous and discrete event dynamics, it can be seen as a hybrid system [4]. Indeed robotic systems have been represented as hybrid systems in different frameworks, even for verification of safety and reachability properties, as in [5]. In [6] a surgical robot is modeled using the hybrid automata model of [7] and some reachability properties are verified.

The scope of our work is to find a verification procedure that is able to follow the modular nature of the system under consideration, in order to prove its properties in a compositional way. Hence, we decided to use the framework of Hybrid Input/Output Automata (HIOAs) of [8] to model the surgical robotic system under consideration, due to its well assessed compositionality and verification theory. Verification issues and extensions of HIOAs have been introduced in [9], [10]. The HIOAs have been used for modelling and verification of safety relevant applications, for instance air traffic control [11] [12], vehicle control [13], [14], [15], [16] and bioinformatics [17].

In our work we start from a very simple example of surgical robotic system, where a manipulator has to reach a desired target. The surgical aspects are not faced in this work, because our aim is to prove in a first instance the modularity of the properties of the system under consideration. To this end we will use the invariant approach in [9]. Using the compositional theory for HIOAs, we will prove that the properties that are true for the automaton of each component, are also true for the overall system, seen as the composition of the automata. Although the approaches used in this paper are not new, they have never been applied to surgical robots with the aim of verifying the safety properties in a compositional way. We want to stress that the compositionality is very useful in the considered context, since a surgical system interacting with human beings has to be reliable in case of component changes, either due to the performed task or to faults. If we are able to verify the desired performance for each component and derive the overall system performance from components ones, we are able to substitute components following a safe policy.

The paper is structured as follows: in section II we introduce the case study under consideration, in section III the theory of HIOAs is recalled, in section IV this case study is modeled using the HIOAs, in section V the verification procedure is presented.

*This research has been supported by the coordination action EuroSurge (grant no. 288233) funded by the European Commission in the 7th EC framework program.

[1]Department of Informatics, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany {luzie.schreiter@kit.edu, joerg.raczkowsky@kit.edu, woern@kit.edu}

[2] Dipartimento di Informatica, Università di Verona, Strada le Grazie 15 37134 Verona, Italy {marta.capiluppi@univr.it, paolo.fiorini@univr.it}

II. Case Study

We consider a manipulator moving freely in a room. The end effector of the manipulator has to reach a target, that might be a tissue in a surgical context. To design our model we will use a simplified version of the scenario described in [6] and [18]. A robot has to be tracked to have a global control of the movements needed to assure a safe interaction with humans during a surgical operation. Referring to Fig. 1, in a first area A, the end effector is tracked by Tracker 1 (T1), which is a high definition tracker with a high framerate. The figure represents the position of the end effector in $\mathbb{R}^2$. The tracker has a small sample time T_{s1} and can follow the marker on the end effector precisely. The manipulator can, then, move following a fast trajectory, until it reaches a point c in the room, where the tracker is changed to Tracker 2 (T2). This second tracker has a bigger sample time T_{s2} and it is less accurate. Hence the manipulator is forced to slow down, following a slower trajectory. This might also be due to the end effector approaching the target.

The real setup is represented in Fig. 2, where a Staubli Puma 260 manipulator is used: it is controlled by a Galil 4080 control board and tracked using Naturalpoint Optitrack (Altair robotic laboratory setup) for Tracker 1 and Claron-Tech Microtracker 2 for Tracker 2.

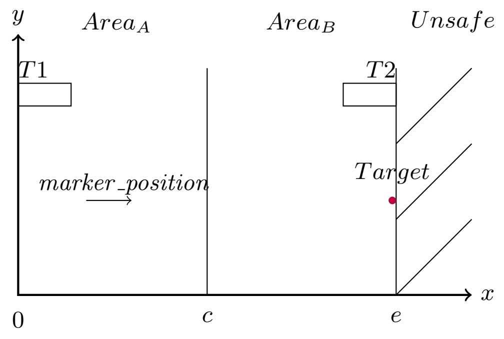

Fig. 1. Schema of the Case Study

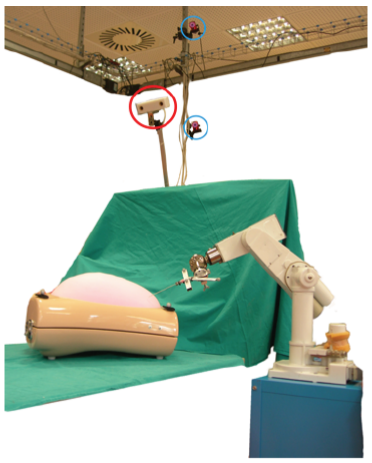

Fig. 2. Possible setup of the case study. T1 and T2 are represented by the circles: blue for T1 and red for T2.

We aim at studying a scenario where the robot moves freely in the surgical room during the setup procedure and its position is monitored by a tracker that controls the entire room. Once the robot approaches the patient we could use a smaller tracker near the end effector that controls only the patient body area during the surgical task. The reason why we use two different trackers is that when the robot moves freely around the room, it has to avoid obstacles and we need a tracker that is able to 'see' the entire room. The second tracker, even if less powerful, is smaller and can occupy less space, hence it can be used near the end effector to track its position next to the patient. Moreover, while the end effectors approaches the target (patient), the speed slows down in any case and the frame rate does not necessarily need to be high. The point in the space and the time in which the tracker is changed can be decided by the user a priori.

III. Recalling Hybrid I/O Automata

For the sake of completeness, we report here some definitions about HIOAs that will be used in the following. A more detailed description of HIOAs can be found in [8].

A HIOA $\mathcal{A}$ is a tuple $((U,X,Y),(I,H,O),Q,D,\Theta,\mathcal{T})$ where

- (U,X,Y) are disjoint sets of input, internal and output variables. Let V denote the set $U \cup X \cup Y$ of variables.
- (I,H,O) are disjoint sets of input, hidden and output actions. Let A denote the set $I \cup H \cup O$ of actions.
- $Q \subseteq vals(X)$ is the set of states
- $\Theta \subseteq Q$ is a non-empty set of start states
- $D \subseteq vals(X) \times A \times vals(X)$ is the discrete transition relation where a is *enabled* in x if there exits a x' such that $x \xrightarrow{a} x'$ with $x \in vals(X)$.
- $\mathcal{T}$ is a set of trajectories on V that satisfy the following axioms:

  T1 Prefix closure

  For every $\tau \in \mathcal{T}$ and every $\tau' \leq \tau, \tau' \in \mathcal{T}$

  T2 Suffix closure

  For every $\tau \in \mathcal{T}$ and every $t \in dom(\tau), \tau \trianglerighteq t \in \mathcal{T}$

  T3 Concatenation closure

  Let $\tau_0, \tau_1, \tau_2, \ldots$ be a sequence of trajectories in $\mathcal{T}$ so that, for each nonfinal index i, τ_i is closed an $\tau_i.lstate = \tau_{i+1}.fstate$. Then $t_0 \frown t_1 \frown t_2 \cdots \in \mathcal{T}$

For each variable v, we assume both a *(static) type*, *type*(v), which gives the set of values it may take on, and a *dynamic type*, *dtype*(v), which gives the set of trajectories it may follow. A *valuation* $\mathbf{v}$ for a set of variables V is a function that associates with each variable $v \in V$ a value in *type*(v). Let J be a left-closed interval of T (the time axis) with left endpoint equal to 0. Then a J*-trajectory* for V is a function $\tau : J \to vals(V)$, such that for each $v \in V$, $\tau \downarrow v \in dtype(v)$. A *trajectory* for V is a J-trajectory for V, for any J. Trajectory τ is a *prefix* of trajectory τ', denoted by $\tau \leq \tau'$, if τ can be obtained by restricting τ' to a subset of its domain. We define $\tau \trianglerighteq t \triangleq (\tau \lceil [t, \infty)) - t$. The concatenation $\frown$ of two trajectories is obtained by taking the union of the first trajectory and the function obtained by shifting the domain of the second trajectory until the start time agrees with the limit time of the first trajectory; the last valuation of the first trajectory, which may not be the same as the first valuation of the second trajectory, is the one that appears in the concatenation. Prefix, suffix and concatenation operations return trajectories. We define $\tau.fval$, the *first valuation* of τ, to be $\tau(0)$, and if τ is closed (J is a closed interval), we define $\tau.lval$, the *last valuation* of τ, to be $\tau(\tau.ltime)$. Given a trajectory $\tau \in \mathcal{T}$ we denote $\tau.fval \lceil X$ by $\tau.fstate$ and, if τ is closed, we denote $\tau.lval \lceil X$ by $\tau.lstate$. We write $f \lceil P$ for the restriction of function f to set P, that is, the function g with $dom(g) = dom(f) \cap P$ such that $g(c) = f(c)$ for each $c \in dom(g)$. If f is a function whose range is a set of functions and P is a set, then we write $f \downarrow P$ for the function g with $dom(g) = dom(f)$ such that $g(c) = f(c) \lceil P$ for each $c \in dom(g)$.

An execution fragment of $\mathcal{A}$ is a sequence $\alpha = \tau_0, a_0, \tau_1, a_1, \tau_2, a_2 ....$ where $a_i \in A$ is an action, $\tau_i \in \mathcal{T}$ is a trajectory and if τ_i is not the last trajectory in α then $\tau_i.lstate \xrightarrow{a_{i+1}} \tau_{i+1}.fstate$. An execution fragment is closed if the final trajectory is a finite closed interval. A state of $\mathcal{A}$ is reachable if it is the last state of a closed execution of $\mathcal{A}$. Thus an execution fragment α is reachable if $\alpha.fstate$ is reachable.

A property of a HIOA $\mathcal{A}$ is a boolean derived variable. A property of $\mathcal{A}$ is stable if the property is true at a certain state ψ and in all reachable states from ψ. A property of $\mathcal{A}$ is invariant if it is stable for all initial states of A [17].

IV. Modelling the case study with HIOAs

As presented in Section 2, the system is composed of the two trackers following their markers on the robot end effector and returning the position of the end effector itself. This position is used by the controller to design its control law based on a reference trajectory. The control variable (joints torques) is then sent to the robot, which moves the end effector and sends the marker signals to the trackers. Coordination is achieved by a supervisor, who receives the trackers positions and their sample times. With this information the supervisor can check which tracker is working at each instant of time and detect the tracker change (i.e. position c of Fig. 1). Moreover, the supervisor computes the reference trajectory according to the speed limits given by the tracker working at that instant of time and sends it to the controller. The supervisor also has to switch on and off the trackers according to the area where the robot is moving and to notify it to the controller.

Each component is modeled with a HIOA. To describe HIOAs we use a variant of the TIOA language [19], with some extensions for hybrid systems [20].

The tracker is represented by the automaton in Fig. 3. It has an internal variable called ID, as identifier, that is used to distinguish Tracker 1 from Tracker 2. In this way we can use the same model for each tracker component, only modifying its ID, and respecting modularity. The tracker input variables are: marker_position, representing the signal sent by the marker positioned on the end effector of the robot; change, representing the signal sent by the supervisor and indicating that the robot crossed point c. The tracker returns the position p_e of the end effector with a delay given by its sample time T_s. The position is stored in an internal variable p and is calculated using the sample time related to each tracker. The action SAMPLE is used to assign to variable p the value of marker_position when a sampling instant is reached. To this end, internal variable t_s keeps track of the elapsing time, and when its value reaches the sample time T_s, action SAMPLE arises and t_s is reset to 0. When the action SWITCH arises the ID is changed from 1 to 2. This occurs when the supervisor sends the variable change, which is stored in an internal variable c. In our model the only parameter that changes due to the 'switch' action is the sample frequency, which goes from f_{s1} to f_{s2}. Indeed, based on this parameter, the position and the current sample time are calculated.

```
hioa Tracker
  variables
      input    marker_position: Real, change: Boolean
      internal $p$: Real := 0, $t_s$: Real := 0,
               $c$: Boolean := 0, ID: $\{1, 2\}$:=1
      output   $p_e$: Real, $T_s$: Real
  actions
      internal SWITCH, SAMPLE
  transitions
      SWITCH
      pre  $c = 1$
      eff  ID := 2
      SAMPLE
      pre  $t_s = T_s$
      eff  $p(t)$ := marker_position$(t)$, $t_s := 0$
  trajectories
      $c(t)$   := change$(t)$;
      $p_e(t)$ := $p(t)$ ;
      $\dot{t}_s(t)$ := 1;
      $T_s(t)$ := $\begin{cases} \frac{1}{f_{s1}} & \text{if ID} = 1 \\ \frac{1}{f_{s2}} & \text{if ID} = 2 \end{cases}$
```

Fig. 3. HIOA of the component: Tracker

```
hioa Robot
  variables
      input    $u$: Real
      internal $q$: Real, $x$: Real
      output   marker_position: Real
  trajectories
      $M(q)\ddot{q}(t) := u(t)$;
      $x(t) := k(q(t))$;
      marker_position$(t) := x(t)$.
```

Fig. 4. HIOA of the component: Robot

Automaton in Fig. 4 represents the Robot (or manipulator).

In this HIOA no action is present. The input variable is the control variable u, given by the joints torques. The manipulator sends the marker signal (variable marker_position) to the trackers monitoring the end effector position. The dynamical law is given by the simplified version of the robot dynamics $M(q)\ddot{q}+C(q,\dot{q})\dot{q}+F(q,\dot{q})+G(q) = u-J^T(q)h$ where q is the vector of generalized coordinates, related to the position and orientation of the end effector by the direct kinematics function $x = k(q)$.

```
***hioa Controller***
  **variables**
        **input**     $(p_e, x_d, \dot{x}_d, \ddot{x}_d)$: Real, change: Boolean
        **internal**  $(u_{int}, x_c)$:Real, changepar:Boolean:=0,
                      $c$: Boolean := 0
        **output**    $u$: Real
  **actions**
        **internal**  NEWPAR
  **transitions**
        NEWPAR
        **pre**   $c = 1$
        **eff**   changepar := 1
  **trajectories**
      $c(t) :=$ change$(t)$;
      $x_c(t) := p_e(t)$;
      $u_{int}(t) := \begin{cases} M_s\ddot{x}_d(t) + D_s(\dot{x}_d - \dot{x}_c)(t) + K_s(x_d - x_c)(t) \\ \text{if changepar = 1} \\ M_f\ddot{x}_d(t) + D_f(\dot{x}_d - \dot{x}_c)(t) + K_f(x_d - x_c)(t) \\ \text{if changepar = 0} \end{cases}$
      $u(t) := u_{int}(t)$
```

Fig. 5. HIOA of the component: Controller

The Controller automaton, represented in Fig. 5, receives as input the position p_e sent by the tracker, the estimated trajectory given by variables $x_d, \dot{x}_d, \ddot{x}_d$ sent by the supervisor, and the change variable, from the supervisor, indicating when the tracker switching has occurred and the controller has to change its parameters accordingly. Indeed when the change variable is received and active (= 1) action NEWPAR arises and the internal variable changepar is set to 1. In this case the tracker is the slowest one, and the parameters of the controller are M_s, D_s, K_s. We used here the model of control law for the free movement. Variable changepar is initialized to 0, indicating that at the beginning the active tracker is Tracker 1, and the used parameter for the control variables are M_f, D_f, K_f.

The Supervisor is described by the automaton in Fig. 6. Its input variables are the end effector position p_e and the tracker sample time T_s sent by the active tracker. Input variable p_e is stored into internal variable p^*. The supervisor compares p^* with the change position c (which is know a priori): when it is reached, i.e. $p^* = c$ the supervisor arises

```
***hioa Supervisor***
  **variables**
        **input**     $(p_e, T_s)$: Real
        **internal**  $\gamma$: Trajectory, (fast, slow): Boolean,
                      $p^*$: Real := 0
        **output**    $(x_d, \dot{x}_d, \ddot{x}_d)$: Real,
                      change: Boolean := 0
  **actions**
        **internal**  TRACKER_SLOW, TRACKER_FAST
  **transitions**
        TRACKER_SLOW
        **pre**   $p^* = c$
        **eff**   slow := 1, change := 1;

        TRACKER_FAST
        **pre**   $p^* \leq c$
        **eff**   fast := 1;
  **trajectories**
        $p^*(t) := p_e(t)$;
        $\gamma(t) := \begin{cases} \gamma_f(p_e(t), T_s(t)) & \text{if fast} \\ \gamma_s(p_e(t), T_s(t)) & \text{if slow} \end{cases}$
        $[x_d(t)\ \dot{x}_d(t)\ \ddot{x}_d(t)] := f(\gamma(t))$
```

Fig. 6. HIOA of the component: Supervisor

action TRACKER_SLOW indicating that Tracker 2 becomes active and Tracker 1 becomes inactive. Hence variables slow and change are set to 1. Variable change is then sent to the Tracker and the Controller. Variables slow and fast are used by the supervisor to know which tracker is active at each instant of time and to calculate the right trajectory: fast movement for Tracker 1 and slow movement for Tracker 2. The action indicating that Tracker 1 is active is TRACKER_FAST and it is active when $p^* \leq c$, setting internal variable fast to 1. The trajectories are calculated using function γ_f for fast movement and γ_s for slow movement and stored in internal variable γ. These functions are not better specified because it is out of the scope of this work. The same for function f used to compute variables $x_d, \dot{x}_d, \ddot{x}_d$ from the trajectory γ and sent to the Controller.

V. Model verification

In this section we present a verification procedure based on invariants (see [9]) to verify that our system is able to change the tracker after position c is crossed. As an overall property for the system we want that the component Tracker is correctly changed when the end effector reaches a predefined point in the space. In this way the system will be able to perform its task even after the component substitution. Hence, we are interested in verifying that before position c Tracker 1 (and only Tracker 1) is active, while after position c Tracker 2 (and only Tracker 2) is active.

To this end, we start from verifying that each component in the system behaves correctly with respect to the global requirements. Hence we prove some invariants on the components that are related to the composed system invariants, and show that we can obtain the global invariants from the local ones. This is useful to make the verification procedure modular, in the sense that if we change a component, we only have to prove local properties and not the global ones.

To this end, some assumptions on the executions of the overall system must be made.

Since the robot has to reach the target point, we assume that point c is crossed only one time and that the reference trajectories for the end effector (both slow and fast) respect this assumption. Point c is constant and it is > 0. We assume that the robot starts from position 0, i.e. at time $t = 0$ the Robot is still and $x(0) =$ marker_position$(0) = 0$. The other initial conditions are $T_s(0) = T_{s1}$, ID = 1, change = 0, $p(0) = p^*(0) = p_e(0) = 0$, $c(0) = 0$. Note that all the other internal variables will have a value determined by the internal initial conditions and by the trajectories of each automaton. When these are not specified, the value of these variables is not interesting.

A consideration due to the delay given by the tracking, is the fact that at instant t^* in which the end effector crosses point c the value of $x(t^*) =$ marker_position(t^*) becomes c. Nevertheless this value is not immediately transmitted to the Supervisor, since it receives the value of p_e delayed by Tracker 1. So basically p_e will take value c when $x(t^*)$ =marker_position$(t^*) = c + \epsilon$. The constant ϵ is due to the delay of transmission of the end effector position which is present between the marker_position and variable p_e, calculated using the sample time of the tracker. Indeed between point c and point $c + \epsilon$ Tracker 1 is still active (as we will prove using the invariants). Tracker 2 is activated when $p_e = c$, i.e. after a delay due to f_{s1}. It is possible to calculate ϵ if we consider the maximum speed of the reference trajectory for Tracker : if we call this speed $\dot{x}_{max}$, then we have $\epsilon = \dot{x}_{max} \times 1/f_{s1}$. Nevertheless, to have an exact perception of what happens between c and $c + \epsilon$ it is sufficient to define numerically the delays due to the trackers and the two reference trajectories.

We are indeed not interested in what happens between c and $c + \epsilon$ since we only want to prove that before c Tracker 1 is active, and after c Tracker 2 is active, i.e. the switching between the two tracker has occurred at some time.

We now prove some invariants on the different components and show that the invariants of the overall composed system can be obtained by composition of the invariants on components.

Note that the system has other invariants in which we are not interested to the scope of this work. We start from the properties on the components Tracker and Supervisor, that are the only two components that are involved in the final task verification. Indeed we want to prove the following global invariants:

P6 = (marker_position $< c \Rightarrow$ ID=1)
P7 = (marker_position $> c + \epsilon \Rightarrow$ ID=2)
with $\epsilon = \dot{x}_{max} \times 1/f_{s1}$

We prove the invariants by induction on the length of a closed execution of Supervisor, following the algorithm presented in [9].

P1 = ($p_e < c \Rightarrow$ change= 0) is an invariant for HIOA presented in Fig. 6. This invariant is extended with $p_e(t) = p^*(t)$. It is true for the initial state, since $p_e(0) = 0$, $c > 0$ and change = 0. For the discrete part, the only action for which precondition is $p_e < c$ is TRACKER_FAST and this action does not affect variable change, which keeps its initial value 0, so the state does not change for the considered variables. For the continuous part, since the variable p_e and change are not internal variables, the invariant does not affect the state of the system.

P2 = ($p_e \geq c \Rightarrow$ change= 1) is an invariant for HIOA presented in Fig.6. It is possible to define the property P2 as $C \vee \widehat{C}$ with $C = (p_e = p^* \geq c \wedge$change = 1) and $\widehat{C} = (p_e = p^* < c)$. From the initial state $p_e(0) = p^*(0) = 0$, $c > 0$ it follows that $\widehat{C}$ is true at the initial state. If we assume that p_e will reach c in an unknown time t (because it is the aim of the surgical task to reach the target point and this point is after c) and under the assertion that $p_e(t) = p^*(t)$ we can assume that C will be true at some state s when p_e reaches c. For the discrete part, we assumed that point c is crossed only once, and the action TRACKER_SLOW is arisen when $p_e = p^* = c$ causing the variable change to take value 1. After point c no other action occurs that can affect the value of variable change, so the state for $p_e = p^* > 0$ is always the same. For the continuous part, since the variable p_e and change are not internal variables, the invariant does not affect the state of the system.

P3 = (change= 0 $\Rightarrow$ ID=1) is an invariant for HIOA presented in Fig. 3. This invariant is extended with change $= c(t)$. It is true for the initial state, since change $= c(0) = 0$ and ID $=$ 1. For the discrete part, the only action of automaton Tracker which has influence on the variable ID is SWITCH, with precondition change $= c(t) = 1$, so it is not related to the considered invariant which keeps the state as the initial state. For the continuous part, since the internal variable ID does not change trajectory due to the continuous evolution, it will keep its previous state.

P4 = (change= 1 $\Rightarrow$ ID=2) is an invariant for HIOA presented in Fig. 3. Since change$(t) = c(t)$ at each instance of time, it is possible to define the property P4 as $B \vee \widehat{B}$ with $B =$ (change$(t) = c(t) = 1 \wedge ID = 2$) and $\widehat{B} =$ (change $\neq 1$). The variable change is of type boolean, thus, based of the initial state, the variable $\widehat{B}$ is true because if $change \neq 1$ the variable change is 0. From invariant P2, the variable change becomes 1 in some state, and change=1 is precondition of action switch that provokes ID to become equal to 2. Hence B is valid in that state. For the discrete part, since the only action affecting ID is SWITCH, whose precondition is $c = 1$, ID keeps its previous value, i.e. ID=2. For the continuous part, since the internal variable ID does not change trajectory due to the continuous evolution, it will keep its previous state.

P5 = ($t_s = T_s \Rightarrow p_e =$ marker_position) is an invariant for HIOA presented in Fig. 3. The initial state of the automaton Tracker is (for the above assertion) $t_s(0) = 0$, and $p_e(0) =$

$p(0)$, and variable t_s increase its value linearly (trajectories) until it reaches the value of T_s either if ID is 1 or 2. When $t_s = T_s$ action SAMPLE arises and variable p changes its value to the value of the input variable marker_position. Since $p_e(t) = p(t)$ for all time instants t, there will be at least one state when P5 is satisfied. For the discrete part, the only action arising in automaton Tracker which has influences on p_e is SAMPLE. Since it arises every time that t_s reaches the value of T_s and its effects are to reset the value of t_s to 0 and to assign to p the value of marker_position, P5 is still valid, keeping in mind that $p_e(t) = p(t)$. For the continuous part, the internal variable p does not change trajectory due to the continuous evolution, it will keep its previous state under the assumption that $p(t)$ =marker_position.

In the next steps transitivity is used to prove invariants on the overall composed system.

P6 = (marker_position $< c \Rightarrow$ ID $= 1$) is an invariant for composition: P1$\wedge$P3$\wedge$P5$\Rightarrow$P6.

P7 = (marker_position $> c + \epsilon \Rightarrow$ ID $= 2$) with $\epsilon = \dot{x}_{max} \times 1/f_{s1}$ is an invariant for composition: P2$\wedge$P4$\wedge$P5$\Rightarrow$P7.

VI. CONCLUSIONS

In this paper we presented a simple case study to introduce a modular verification procedure for surgical robotic systems. Although the case study is simplified and no surgical aspect is taken under consideration, the main aim of this work is to propose a way of decomposing a robotic system (with a surgical aim in our mind) following its components. Since the surgical context needs procedures for verifying safety, we chose to present a system where a component can be changed without losing the performance of the system, which is very important in a surgical context. To this end we decided to adopt the formalism of Hybrid I/O Automata of [8] to model the system in a modular way and take advantage of the compositionality theory behind this model to verify the overall system properties starting from each component property. In the future we aim at extending these results including some typical man-machine interaction issues, due to the surgical context. Moreover, more complex techniques, including simulation and constraint-based methods, will be applied to the same kind of systems, generalizing the results to a class of surgical robotic scenarios.

ACKNOWLEDGMENTS

The authors would like to thank Prof. Roberto Segala for his help with the theory of HIOAs and Fabrizio Boriero for his help in creating the real set-up of Fig. 2.

REFERENCES

[1] K. Cleary and C. Nguyen, "State of the art in surgical robotics: clinical applications and technology challenges," *Computer Aided Surgery*, vol. 6, no. 6, pp. 312–328, 2001.

[2] M. Althoff, D. Althoff, D. Wollherr, and M. Buss, "Safety verification of autonomous vehicles for coordinated evasive maneuvers," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, June 2010, pp. 1078 –1083.

[3] E. Mitka, A. Gasteratos, N. Kyriakoulis, and S. G. Mouroutsos, "Safety certification requirements for domestic robots," *Safety Science*, vol. 50, no. 9, pp. 1888 – 1897, 2012.

[4] 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," *Theoretical Computer Science*, vol. 138, pp. 3–34, 1995.

[5] J. Ding, J. Gillula, H. Huang, M. Vitus, W. Zhang, and C. Tomlin, "Hybrid systems in robotics: Toward reachability-based controller design," *Robotics Automation Magazine, IEEE*, vol. 18, no. 3, pp. 33 –43, sept. 2011.

[6] R. Muradore, D. Bresolin, L. Geretti, P. Fiorini, and T. Villa, "Robotic surgery: Formal verification and plans," *Robotics Automation Magazine, IEEE*, vol. 18, no. 3, pp. 24 –32, sept. 2011.

[7] R. Alur, C. Courcoubetis, T. A. Henzinger, and P. H. Ho, "Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems," in *Hybrid Systems*, 1992, pp. 209–229.

[8] N. Lynch, R. Segala, and F. Vaandrager, "Hybrid i/o automata," *Information and Computation*, vol. 185, no. 1, pp. 105 – 157, 2003.

[9] S. Mitra and D. Liberzon, "Stability of hybrid automata with average dwell time: an invariant approach," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 2, dec. 2004, pp. 1394 – 1399 Vol.2.

[10] M. Capiluppi and R. Segala, "Modelling implicit communication in multi-agent systems with hybrid input/output automata." in *Proceedings of the Third International Symposium on Games, Automata, Logics and Formal Verification (GandALF 2012)*, Naples, Italy, September 2012.

[11] J. Lygeros and N. Lynch, "On the formal verification of the tcas conflict resolution algorithms," in *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, vol. 2, dec 1997, pp. 1829 –1834 vol.2.

[12] C. Livadas, J. Lygeros, and N. Lynch, "High-level modeling and analysis of the traffic alert and collision avoidance system (tcas)," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 926 –948, july 2000.

[13] H. Weinberg and N. Lynch, "Correctness of vehicle control systems-a case study," in *Real-Time Systems Symposium, 1996., 17th IEEE*, dec 1996, pp. 62 –72.

[14] J. Lygeros and N. Lynch, "Strings of vehicles: Modeling safety conditions," in *Proceedings of the First International Workshop on Hybrid Systems*. Springer-Verlag, 1998, pp. 273–288.

[15] E. Dolginova and N. Lynch, "Safety verification for automated platoon maneuvers: A case study," in *Proc. of HART, LNCS 1201*, 1997, pp. 154–170.

[16] E. Marinica, M. Capiluppi, J. Rogge, R. Segala, and R. Boel, "Distributed collision avoidance for autonomous vehicles: world automata representation," in *4th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS)*, 2012.

[17] K. Koutroumpas and J. Lygeros, "Modeling and verification of stochastic hybrid systems using hioa: a case study on dna replication," in *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, ser. HSCC '10. New York, NY, USA: ACM, 2010, pp. 263–272.

[18] D. Bresolin, L. Di Guglielmo, L. Geretti, R. Muradore, P. Fiorini, and T. Villa, "Open problems in verification and refinement of autonomous robotic systems," in *Digital System Design (DSD), 2012 15th Euromicro Conference on*, 2012, pp. 469–476.

[19] D. Kaynar, N. Lynch, R. Segala, and F. Vaandrager, "The theory of timed i/o automata," *Synthesis Lectures on Computer Science*, 2006.

[20] S. Mitra, Y. Wang, N. Lynch, and E. Feron, "Safety verification of model helicopter controller using hybrid input/output automata," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, O. Maler and A. Pnueli, Eds., vol. 2623. Springer-Verlag, Berlin, 2003, pp. 343–358.