Formal Verification of robotic surgery tasks by reachability analysis

Davide Bresolin[a], Luca Geretti[b], Riccardo Muradore[c], Paolo Fiorini[c], Tiziano Villa[c]

[a] *University of Bologna (Italy)*
[b] *University of Udine (Italy)*
[c] *University of Verona (Italy)*

Abstract

In this paper we discuss the application of formal methods for the verification of properties of control systems designed for autonomous robotic systems. We illustrate our proposal in the context of surgery by considering the automatic execution of a simple action such as puncturing. To prove that a sequence of subtasks planned on pre-operative data can successfully accomplish the surgical operation despite model uncertainties, we specify the problem by using hybrid automata. We express the requirements of interest as questions about reachability properties of the hybrid automaton model. Then, we use the tool Ariadne to study how the choice of the control parameters and the measurement error affect the safety of the system.

Keywords: Formal Verification, Hybrid Systems, Surgical Robotics

1. Introduction

In the last decades robotics played a relevant role in the progress of surgery. Even though *robotic surgery* has been a new field of investigation, in a short time the prototypes built in robotics laboratories gained a place into operating rooms. A new terminology witnesses this trend: computer-integrated surgery, medical robotics, rehabilitation robotics, telerobotics, telesurgery, robotic assistive systems, robot-assisted laparoscopic surgery, etc. [1, 2, 3]. Robotic surgery has already proven its advantages by improving safety, accuracy, reproducibility, and decreasing patient recovery time and surgeon fatigue [4, 5, 6, 7]. This is both a blessing and a curse, in the sense that the advanced functionalities can be useful for surgeons, but at the same time the increased complexity makes the system more susceptible to design errors and poses new challenges to the

Email addresses: `davide.bresolin@unibo.it` (Davide Bresolin), `luca.geretti@uniud.it` (Luca Geretti), `riccardo.muradore@univr.it` (Riccardo Muradore), `paolo.fiorini@univr.it` (Paolo Fiorini), `tiziano.villa@univr.it` (Tiziano Villa)

designers. Advanced design, control, monitoring and deployment paradigms are needed for the next-generation robotic surgery sytems.

In engineering practice, the analysis of a complex system is usually carried out via simulation, which allows the designers to explore one of the possible system executions at a time. Formal methods instead aim at exploring all possible executions, in order to ensure proper functionality of the system in all cases, or conversely to acquire information about potential fault cases. In computer science, the name Formal Methods identifies a large family of mathematical languages, techniques and tools used to specify and verify systems and to help engineers to develop more reliable systems. Nowadays, they are standard practice in many ICT industries for the development of (discrete) HW/SW systems, and are becoming a vital aspect in the design of safety-critical cyberphysical systems, including robotics and automation systems [8, 9]. An area where they can greatly improve the reliability of the design process is Autonomous Robotic Surgery (ARS) [10, 11]. The aim of ARS is to perform simple tasks without the presence (or telepresence) of surgeons. Therefore, with ARS, basic tasks will be executed by robots, allowing the surgeons to focus only on the most difficult aspects of the intervention. This implies that the overall control architecture must respect strict safety constraints and must guarantee the successful accomplishment of the surgical tasks, independently of uncertainties and un-modeled subsystems.

In this work we will show how formal verification can provide accurate and reliable answers to help the designer in the development of ARS systems by considering the automatic execution of a simple surgical action such as puncturing. We first model the overall task as a finite sequence of atomic actions that should be accomplished to guarantee the success of the surgical action. This model takes the form of a *hybrid automaton* consisting of a discrete control part that operates in a continuous environment [12]. Then, we specify the safety constraints that the system should respect in a precise mathematical way as reachability properties of the hybrid automaton model. Finally, we use a state-of-the art tool for reachability analysis of hybrid automata (Ariadne [13]) to find the values of control parameters that guarantee successful accomplishment of the surgical operation, even in presence of measurement errors. This paper focuses on the the second step of the usual design process: mathematical modeling $\rightarrow$ simulation & formal verification $\rightarrow$ mechanical design $\rightarrow$ experimental validation, and has a twofold goal: (1) to highlight how a formal verification tool can be used to predict the performance of a robotic system, and (2) to help the designer during the tuning phase of the controller. We leave the study on the subsequent design step to forthcoming publications.

The paper extends the preliminary results reported in [14] and is organized as follows. In Section 3 we model the surgical task and provide a mathematical model of the robotic manipulator. Section 4 formally defines the properties to be verified, whereas Section 5 describes the verification strategy and the results of the experiments. Some conclusions are drawn in Section 6.

	PHAVer	SpaceEx	HSOLVER	Ariadne
State space representation	Polyhedra	Support functions	Predicate abstraction	Image sets
Nonlinear dynamics	no	no	**YES**	**YES**
Composition of automata	**YES**	**YES**	no	**YES**
Rigorous results	**YES**	no	**YES**	**YES**
User-definable accuracy	**YES**	**YES**	no	**YES**
Graphical output of results	**YES**	**YES**	no	**YES**
Max. no. of variables*	~ 10	$\sim$ **100**	~ 10	~ 10

Table 1: Comparison of PHAVer, SpaceEx, HSOLVER and Ariadne. (*these numbers of variables were reached in some cases reported respectively in [15], [16], [17] and [13].)

2. Review of Formal Verification Tools

When the system dynamics are simple, the evolution can be computed exactly, and most of the verification techniques for finite-state models can be used to obtain an exact answer to verification problems. When the dynamics is more complex, the reachable set cannot be computed exactly, and different techniques are needed to face the complexity of the verification problem. One of the most successful approaches is to use approximation techniques to under- or over-approximate the evolution of the system.

Among the publicly available state-of-the art tools that use approximation techniques to verify hybrid automata, the most relevant and actively developed are PHAVer [15], SpaceEx [16], HSOLVER [17], and Ariadne [13]. In the following we briefly describe the four tools. We refer the reader to the specific literature for a comprehensive description of their algorithms and state space representation choices. Table 1 summarizes their differences under the following criteria:

- **Class of system they can verify:** do they support nonlinear dynamics? Can the system be specified as a composition of smaller components?
- **Soundness of the results:** is the verification result guaranteed to be mathematically correct?
- **Accuracy control:** is it possible to choose the quality of the approximations?
- **Output:** is it possible to obtain a graphical output of the results, or is only a YES/NO answer provided?
- **Scalability:** what is the maximum size of a system that they can verify?

PHAVer [15] was one of the first tools that enabled verification of hybrid automata with complex dynamics: it handles affine dynamics and guards and supports the composition of hybrid automata. The state space is represented using polytopes. Results are formally sound by means of an exact and robust arithmetic with unlimited precision. Scalability is limited: systems with more than 10 continuous variables are usually out of the capabilities of the tool.

SpaceEx [16] is a modular open-source framework that improves upon PHAVer, with particular regard to scalability (systems with 100 variables have been analyzed with this tool). It combines polyhedra and support functions to represent the state space of systems with piecewise affine, non-deterministic dynamics. Local error bounds on the computation are guaranteed using variable time steps; however, differently from PHAVer, the result of SpaceEx is not guaranteed to be numerically sound. This means that when the tool finds the system safe, we can only conclude that more sophisticated methods are necessary to find bugs for that system.

HSOLVER [17] uses constraint propagation and abstraction-refinement techniques to verify safety properties of nonlinear hybrid systems. In this setting, the hybrid system under verification is first abstracted by a finite-state discrete model that approximates the original one. If the abstraction is not accurate enough to obtain an answer to the verification problem, it is improved using constraint propagation techniques, until either an answer is found or the maximum number of refinement steps is reached. HSOLVER supports systems with complex non-linear dynamics and guards, but not the composition of automata. Because of the particular state-space representation, it cannot provide a graphical output of the reachable set, but only a safe/possibly unsafe answer to the verification problem.

Ariadne [13] uses rigorous numerical methods for working with real numbers, functions and sets in the Euclidean space, to verify hybrid systems with nonlinear dynamics, guards and reset functions. It supports composition to build complex systems from simpler components, and can compute both upper-approximations and lower-approximations of the reachable set. By combining outer and lower approximations, Ariadne can provide both positive and negative answers to safety properties and other more complex verification problems. Its high expressivity, however, affects the performance and scalability of the tool, which is currently limited to systems with 10 continuous variables.

3. Modeling a Surgical Task

Puncturing is the act of penetrating a biological tissue with a needle, e.g. when performing a biopsy. Together with other elementary tasks such as cutting and suturing, this action can be used to build more complex surgical tasks. To model the puncturing task in a formal way, we divided its execution into three subtasks: *(i)* a *free motion* phase, where the end effector of the robot approaches the patient's tissue starting from its home position; *(ii)* a *perpendicular attitude* phase, where the end effector is in contact with the tissue, and the robot moves its wrist to have the tool orthogonal with the patient's surface; *(iii)* a *puncturing* phase, where the robot increases the force applied by the end effector until the tissue is penetrated.

We assume that the controller for each subtask stabilizes the plant, while the switching between controllers preserves the stability. Our goal is not to prove the stability of the overall system but to prove in a formal way that the *task* itself can be executed correctly. Thus, the focus of the analysis is to show

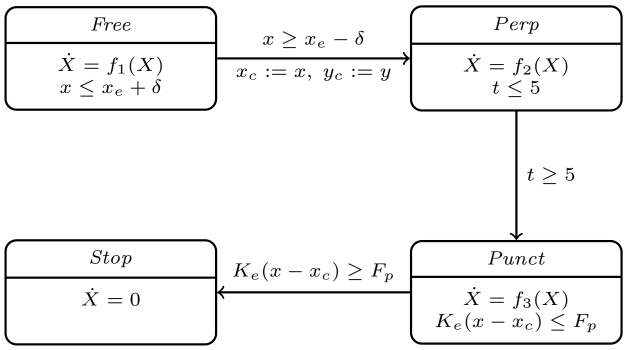

Figure 1: Automaton model of the surgical test bench.

the feasibility of the task rather than the stability of the system. The test case under consideration is a typical example of a *hybrid system*, i.e., a system mixing discrete and continuous behaviour that cannot be characterized faithfully using either discrete or continuous models only. A hybrid system consists of a discrete part that operates in a continuous environment, and for this reason it is sensitive not only to time-driven phenomena but also to event-driven phenomena.

We model the surgical task by using the well known formalism of hybrid automata [12]. Intuitively, a hybrid automaton is a "finite-state automaton" with continuous variables that evolve according to dynamics specified at each discrete node.

Definition 1. *A* hybrid automaton *is a tuple* $\mathcal{A} = \langle \mathrm{Loc}, \mathrm{Edg}, X, \mathrm{Inv}, \mathrm{Dyn}, \mathrm{Act}, \mathrm{Res} \rangle$ *such that:*

1. $\langle \mathrm{Loc}, \mathrm{Edg} \rangle$ *is a finite directed graph; the vertices,* Loc*, are called* locations *or* control modes*, and the directed edges,* Edg*, are called* discrete transitions *or* control switches*;*

2. X *is a finite set of* continuous variables *representing the continuous state space of the system;*

3. *each location* $\ell \in \mathrm{Loc}$ *is labeled by the* invariant condition $\mathrm{Inv}[\ell]$ *on* X *and the* dynamic law $\mathrm{Dyn}\,[\ell]$ *on* $X \times X \times \mathbb{R}^{\geq 0}$ *such that if* $\mathrm{Inv}[\ell](\mathbf{x})$ *is true then* $\mathrm{Dyn}\,[\ell](\mathbf{x}, \mathbf{x}, 0)$ *is true;*

4. *each edge* $e \in \mathrm{Edg}$ *is labeled by the* activation condition $\mathrm{Act}\,[e]$ *on* X *and the* reset relation $\mathrm{Res}\,[e]$ *on* $X \times X$*.*

A *state* of a hybrid automaton $\mathcal{A}$ is a pair $\langle \ell, \mathbf{x} \rangle$, where $\ell \in \mathrm{Loc}$ is a location and $\mathbf{x}$ is an assignment of values for the continuous variables in X. A state $\langle \ell, \mathbf{x} \rangle$ is said to be *admissible* if $\mathrm{Inv}[\ell](\mathbf{x})$ holds.

The evolution of a hybrid automaton alternates *continuous* and *discrete* steps. In a continuous step, the location does not change, while the continuous variables change following the continuous dynamics $\mathrm{Dyn}\,[\ell]$ of the location. A discrete evolution step consists of the activation of a discrete transition

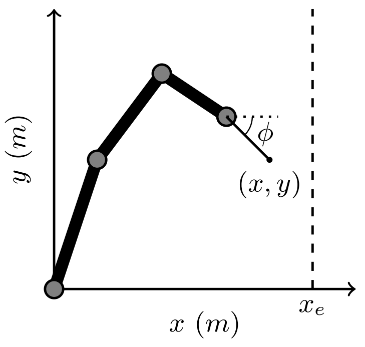

Figure 2: Cartesian state space of the surgical test bench.

$e \in \mathrm{Edg}$ that can change both the current location and the value of the state variables, in accordance with the reset function $\mathrm{Res}\,[e]$ associated to the transition. The interleaving of continuous and discrete evolutions is decided by the invariant $\mathrm{Inv}[\ell]$ of the location, which must be true for the continuous evolution to keep on going, and by the guard predicate $\mathrm{Act}\,[e]$, which must be true for a discrete transition e to be activated. Formally, a *trajectory* ξ of a hybrid automaton can be defined as a (finite or infinite) sequence $(\xi_i)_{i\geq 0}$ of continuous functions $\xi_i : [t_i, t_{i+1}] \to \mathrm{Loc} \times X$ such that $\mathrm{Dyn}[\ell](\xi_i(s), \xi_i(t), t-s)$ holds for all $t_i \leq s \leq t \leq t_{i+1}$, and both $\mathrm{Act}\,[e](\xi_i(t_{i+1}))$ and $\mathrm{Res}\,[e](\xi_i(t_{i+1}),\ \xi_{i+i}(t_{i+1}))$ hold for some $e \in \mathrm{Edg}$. Here, $\xi_i(t)$ represents the state of the system after i events and at time t.

In our test case, each location of the automaton corresponds to one of the subtasks identified for the surgical action. The automaton describing a simplified version of the puncturing action is shown in Figure 1. The continuous space is given by the 9-dimensional set of variables

$$X = \{x, v_x, y, v_y, \phi, v_\phi, t, x_c, y_c\}, \tag{1}$$

where x and y represent the position of the end-effector on a plane, ϕ is the orientation of the needle, v_x, v_y and v_ϕ are the first derivatives of x, y, and ϕ, t is the time elapsed and x_c and y_c are auxiliary variables that store the position of the contact point of the end effector with the patient's skin. The patient is assumed to lie on a plane orthogonal to the x coordinate, with an unknown position in the range $[x_e - \delta, x_e + \delta]$, where δ is the uncertainty and $x_e = 0.95m$ the nominal position.

Locations *Free*, *Perp* and *Punct* correspond to the three subtasks in which the puncturing action is divided, while location *Stop* is reached upon successful execution of the task. Transitions describe the switching from one subtask to another, and are defined as follows:

- the transition from *Free* to *Perp* is activated when the end effector touches the tissue. To model the uncertainty in the position of the patient, this transition can be taken as soon as x is greater or equal to $x_e - \delta$, but the automaton is not forced to leave location *Free* if x is less or equal to $x_e + \delta$. Upon activation of the transition, the actual value of the contact point between the end-effector and the patient is stored in variables x_c and y_c;

- the manipulator remains in location *Perp* while the time is less than $5s$. At $t = 5s$ the tool is perpendicular to the tissue, the transition is activated and the automaton moves to *Punct*;
- the last transition becomes active as soon as the force applied by the needle (given by the expression $K_e(x - x_c)$ assuming that the tissue can be modeled as a spring with stiffness equal to K_e) is greater than the threshold F_p, and the tissue is penetrated.

In location *Stop* the robot is assumed to remain motionless, hence the derivatives of all variables are set to 0. To give the details of the continuous dynamics in locations *Free*, *Perp* and *Punct*, we have to briefly recall the dynamic model of the robot and of the controller. A serial link manipulator in joint space with n-degrees of freedom is described by a set of nonlinear differential equations

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + F(\dot{\mathbf{q}}) + G(\mathbf{q}) = \mathbf{u} - J^T(\mathbf{q})\mathbf{h} \tag{2}$$

where $\mathbf{q} = \begin{bmatrix} q_1 & \cdots & q_n \end{bmatrix}^T$ is the vector of generalized coordinates with corresponding velocity $\dot{\mathbf{q}}$ and acceleration $\ddot{\mathbf{q}}$, $\mathbf{u} = \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix}^T$ is the command torque vector, and $\mathbf{h} = \begin{bmatrix} \mathbf{f}_e^T & \mathbf{u}_e^T \end{bmatrix}^T$ is the force/torque vector applied by the end effector when the robot is in contact with the environment. In this standard Lagrangian representation, M is the symmetric non-singular moment of inertia matrix, C is the Coriolis and centrifugal force matrix, $F(\dot{\mathbf{q}})$ is the frictional torque, and G is the gravitational part. The relation between the external force and the torque at the joint level is given by the transpose of the Jacobian $J^T(\mathbf{q})$ evaluated at the current position $\mathbf{q}$ [18].

Equation (2) can be re-written in Cartesian or task space, where the nominal trajectory is usually designed: in our case, the three-dimensional space $\mathbf{x} = \begin{bmatrix} x & y & \phi \end{bmatrix}^T$ as shown in Figure 2. Using the direct kinematic and the relationship between geometric and analytical Jacobians, we end up with

$$\Lambda(\mathbf{x})\ddot{\mathbf{x}} + \Xi(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \gamma(\mathbf{x}) = \boldsymbol{\tau} + \mathbf{f}_e \tag{3}$$

where $\mathbf{x} = \begin{bmatrix} x & y & \phi \end{bmatrix}^{\mathbf{T}}$ is the pose (position and orientation) of the end effector (i.e. the tip of the needle). The matrices $\Lambda(\mathbf{x})$ and $\Xi(\mathbf{x}, \dot{\mathbf{x}})$ are the transformed inertia matrix and the Coriolis/centrifugal matrix, respectively. The torque $\boldsymbol{\tau}$ is the control vector and $\mathbf{f}_e$ is the vector of generalized interaction force. We refer the reader to [18] for more details and properties of this mathematical representation.

In our example the model of the robot is assumed known, and hence it is possible to implement the inverse dynamics control within the parallel force/position

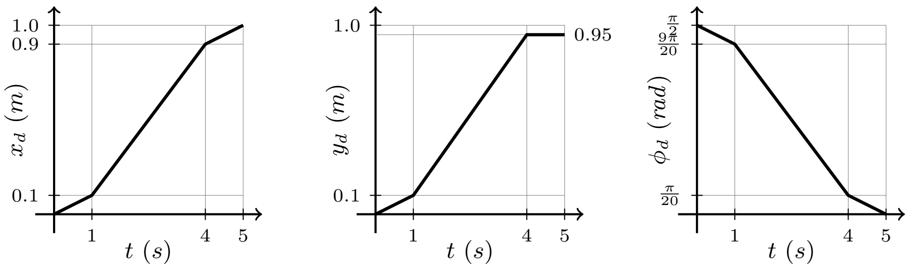

Figure 3: Reference trajectory $\mathbf{x}_d(t)$.

scheme [18]. Let $\mathbf{x}_d - \mathbf{x}$ be the tracking error between the reference trajectory $\mathbf{x}_d$ and the actual Cartesian position $\mathbf{x}$. In free motion ($\mathbf{f}_e = 0$) the robot model coupled with the position control action becomes

$$M_d\ddot{\mathbf{x}} = M_d\ddot{\mathbf{x}}_d + K_D(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + K_P(\mathbf{x}_d - \mathbf{x}), \tag{4}$$

where M_d, K_D, K_P are positive design parameters whose choice is application dependent. When the end effector is in contact with the patient, the right-hand side of the control equation has also a term associated with the interaction force $\mathbf{f}_e = K_e(\mathbf{x} - \mathbf{x}_c)$:

$$M_d\ddot{\mathbf{x}} = M_d\ddot{\mathbf{x}}_d + K_D(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + K_P(\mathbf{x}_d - \mathbf{x}) + K_f K_e(\mathbf{x} - \mathbf{x}_c). \tag{5}$$

where $\mathbf{x}_c$ is the contact point in Cartesian coordinates and K_e is the stiffness matrix of the tissue[1]. The control force action $K_f K_e(\mathbf{x} - \mathbf{x}_c)$ depends on the design parameter K_f. This parameter is a sort of trade off between the position action and the force action: it is in charge of adapting the nominal trajectory $\mathbf{x}_d$ according to the current interaction force.

When the automaton is in locations *Free* and *Perp*, the reference trajectory $\mathbf{x}_d(t)$ starts from $(0, 0, \frac{\pi}{2})$ and ends in $(1.0, 0.95, 0)$ after $5s$ following the piecewise linear functions depicted in Figure 3. In location *Punct* the reference trajectory keeps y and ϕ constant, and increases the value of x to penetrate the tissue:

$$x_d(t) = x_c + \tfrac{1}{2}(t - 5) \qquad y_d(t) = y_c \qquad \phi_d(t) = 0 \tag{6}$$

In all locations the dynamics of variables x, y, ϕ, t, x_c, y_c are fixed and set to $\dot{x} = v_x$, $\dot{y} = v_y$, $\dot{\phi} = v_\phi$, $\dot{t} = 1$, $\dot{x}_c = 0$, $\dot{y}_c = 0$. The dynamics for v_x, v_y and v_ϕ are obtained by substituting $\mathbf{x}_d$ reported in Figure 3 in (4) for location *Free*, in (5) for location *Perp*, and $\mathbf{x}_d$ from (6) in (4) for location *Punct*.

[1]In the experiments, we set $M_d = 100kg$, $K_e = 1000N/m$ and we vary the control parameters

4. Expressing Properties of a Surgical Task

The verification of hybrid systems is usually carried out by reachability analysis, i.e., by computing the set of all states that can be reached under the dynamical evolution starting from a given initial state set. Formally, we say that a state $\langle \ell_r, r\rangle$ *reaches* a state $\langle \ell_s, s\rangle$ if there exists a finite trajectory $\xi = (\xi_i)_{0\le i\le n}$ such that $\xi_0(0) = \langle \ell_r, r\rangle$ and $\xi_n(t_{n+1}) = \langle \ell_s, s\rangle$. We use $\mathit{ReachSet}_{\mathcal{A}}(\langle \ell_r, r\rangle)$ to denote the set of states reachable from $\langle \ell_r, r\rangle$. Moreover, given a set of *initial states* $X_0 \subseteq \mathrm{Loc} \times X$ we use $\mathit{ReachSet}_{\mathcal{A}}(X_0)$ to denote the set $\cup_{\langle \ell_r, r\rangle \in X_0} \mathit{ReachSet}_{\mathcal{A}}(\langle \ell_r, r\rangle)$. The *reachability problem* for hybrid automata is the problem of computing the *reachable set* $\mathit{ReachSet}_{\mathcal{A}}(X_0)$.

Doing verification by reachability analysis implies that tools typically accept a subset of the state space called *safe set* as a specification, and test whether the reachable set is included into the safe set or not. This means that they are limited to *safety properties*, that is, properties that hold for all possible executions and for all time instants. Moreover, the properties cannot be specified using some specification language, but must be transformed into a safe set by the user. Nevertheless, they can still be used to verify meaningful properties of cyberphysical systems.

In the following we concentrate our attention on the following property of the surgical task.

> Property P: *The task is feasible, and the position of the needle at the end of the task is always inside a given target region R.*

This property can be formalized as $\mathit{Eventually}(\mathit{Stop} \wedge \mathbf{x} \in R)$, that unfortunately is not a safety property, since it requires every trajectory of the system to reach the target region at some unspecified time instant. Hence, it cannot be represented by a safe set and thus it cannot be verified by current tools. However, if we impose the task to be completed before a maximum time t_{max} and if we add an auxiliary variable t representing the time elapsed, we can rephrase the property as a safety property:

$$\mathit{Always}(t \ge t_{max} \rightarrow (\mathit{Stop} \wedge \mathbf{x} \in R)),$$

which corresponds to the safe set $\{(\ell, \mathbf{x}) \mid t < t_{max}\} \cup \{(\ell, \mathbf{x}) \mid t \ge t_{max} \wedge \ell = \mathit{Stop} \wedge \mathbf{x} \in R\}$.

In the next section we will study the satisfaction of the Property P with respect to the values of two design parameters, namely, to the proportional and derivative gains K_P and K_D of the controller, and to the measurement errors.

5. Verification of a Surgical Task

Once the task is modeled as described in Section 3 and the properties of interests are formalized as in Section 4, formal verification algorithms can be used to analyze the behavior of the system and obtain an answer to the question "does the system respect the desired properties?". We would like to emphasize that one of the advantages of formal methods is that they are independent of both the specific system and the specific property. Indeed, as long as the system is modeled by an hybrid automaton and properties are formalized as safe sets, any tool of Section 2 can, in principle, obtain an answer to the verification problem. Specific modeling choices like partitioning into sub-tasks, the number of state variables or the complexity of the property can affect the performance of the tools, but do not invalidate the verification methodology.

This high flexibility has a price: unfortunately, the verification of hybrid automata is an undecidable problem. There is no algorithm that can compute the reachable set in an exact way without imposing strong restrictions on the dynamics of the automaton [12]. Nevertheless, approximation techniques can be exploited to verify hybrid automata with complex dynamics: suppose we can compute an over-approximation S to $\mathit{ReachSet}_{\mathcal{A}}(X_0)$, that is, a set $S \supseteq \mathit{ReachSet}_{\mathcal{A}}(X_0)$. Then if S is a subset of the safe set, then so is the reachable set and the automaton $\mathcal{A}$ satisfies the property. Conversely, if we can compute an under-approximation S to $\mathit{ReachSet}_{\mathcal{A}}(X_0)$ (that is, a set $S \subseteq \mathit{ReachSet}_{\mathcal{A}}(X_0)$) that turns out to contain at least one point outside the safe set, we have proved that $\mathcal{A}$ does not respect the safety property φ.

In this section we study the dependence of the satisfaction of the property on the two control parameters K_D and K_P, and on measurement errors. We performed the verification using Ariadne. The choice of the tool was motivated by two reasons. First of all, the tool comparison in [14] (briefly recalled in Section 2) suggests that our puncturing test case is outside the capabilities of tools like PHAVer and HSOLVER, and that the improved reachability algorithms implemented by recent tools are needed. Second, to obtain both a positive and a negative answer to the verification problem we need a tool that is able to compute both over-approximations and under-approximations of the reachable set. Ariadne fulfills both requirements: it uses Images Sets and Taylor expansion approximations to manipulate efficiently functions and sets in the Euclidean space, and can compute not only upper-approximations of the reachable sets, but also *lower-approximations*. Formally, an ε-lower approximation L_ε of a set S is a set such that, for every point $x \in L_\varepsilon$, there exists a point $y \in S$ at a distance less than ε from x. Hence, they can play the role of under approximations: if there exists a point in L_ε at a distance more than ε from the safe set we can conclude that the system is unsafe.

In the first step of this verification example, we determine the values of K_D and K_P for which Property P is true, setting a maximum time to complete the task equal to 10 sec. and a target region such that $y \in [0.94, 0.96]$. We set the range of possible values for K_D as $[100, 8000]$ and a range $[500, 3500]$ for K_P, and we divided the parameter space into a 64×64 grid. Figure 4 shows the results of the verification: green dots are values of the control parameters where Ariadne provided a "safe" answer, red dots correspond to the values of the parameters where the tool provided an "unsafe" answer, while yellow dots depict the values of the parameters for which a definite answer cannot be found.

In the choice of the controller parameters, successful accomplishment of the task is the main objective, but not the only one. Another important feature is

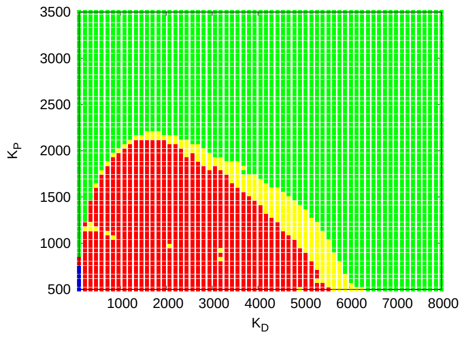

Figure 4: Satisfaction of Property P with no measurement error.

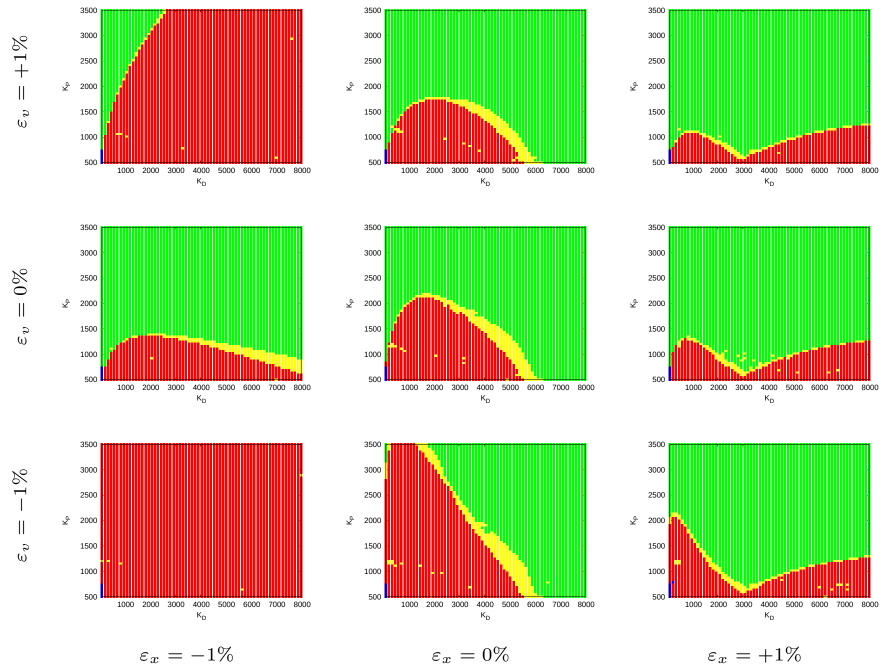

Figure 5: Satisfaction of Property P for different position/velocity measurement errors.

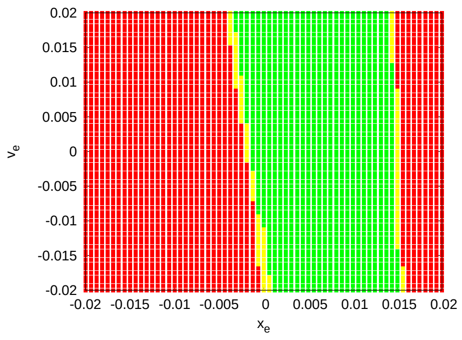

Figure 6: Satisfaction of Property P for $K_D = 4000$, $K_P = 2000$ and different measurement errors.

the robustness of the controller, that is, its ability to achieve the control goal also in presence of errors and noise. Hence, as a second verification example we studied the effect of measurement errors on the robotic surgery example. More precisely, we introduced a constant multiplicative error (i.e. a relative bias) in Equations (4) and (5), by replacing $\mathbf{x}$ and $\dot{\mathbf{x}}$ with $\mathbf{x}_m = \mathbf{x}(1 + \varepsilon_x)$ and $\dot{\mathbf{x}}_m = \dot{\mathbf{x}}(1 + \varepsilon_v)$, where ε_x and ε_v are the position and velocity multiplicative errors, respectively. Figure 5 shows the results of the space analysis of controller parameters for different combinations of position and speed errors.

The following observations can be drawn:

- the completion of the task is strictly related to the combination of errors and not only to their magnitudes, i.e. the case $(\varepsilon_x, \varepsilon_v) = (-1\%, -1\%)$ is much more severe than $(\varepsilon_x, \varepsilon_v) = (+1\%, +1\%)$ even if the amount of bias is the same (in modulus).
- the combination of $(\varepsilon_x, \varepsilon_v)$ affects the set of pairs K_P, K_D that is feasible for the task.
- a safe design should select a pair K_P, K_D that is feasible in any condition, i.e. that belongs to any green zone. In the present case this is not possible because no pairs K_P, K_D satisfy the case $(\varepsilon_x, \varepsilon_v) = (-1\%, -1\%)$: we should enlarge the range for the control parameters and/or change the kind of controller for handling underestimates of position and velocity.

We conclude this section with a finer study of the effect of errors on the successful achievement of the task. We fixed the values of the control parameters to $K_D = 4000$ and $K_P = 2000$, and we determined the values of the measurement errors ε_x and ε_v such that the desired property is satisfied. The range of possible values for the measurement errors is $[-0.02, 0.02]$ (i.e., from -2% to 2%). As in the previous verification runs, the space is divided into a 64×64 grid. Figure 6 shows the results of this last verification example.

6. Conclusions

These experiments prove that formal verification can be useful within the control design cycle to prove the correctness of task execution and to guide the designers in tuning the control parameters. However, the existing tools still lack in general scalability and robustness to be truly practical for control engineers. This motivates the need for further research efforts to improve these features and turn formal verification into a standard practice for the engineering of robotic surgery systems.

One possible way to overcome the limitations of current tools is to develop *abstraction techniques* that can simplify the model of the system to make it tractable without loosing information on the properties of interest [19]. An alternative approach is to exploit *assume-guarantee reasoning* to simplify the verification problems to be fed into the tools. In this approach the system under verification is seen as an assembly of components, represented in terms of assumptions about the environment and guarantees about the components' behavior. In this way, the verification problem for the whole system is decomposed into a set of simpler problems that, if satisfied, guarantee that the overall problem is satisfied [20, 21].

In the future we plan to apply the proposed methodology to a real robotic system, to use the outcomes of the formal verification phase to tune the controller, and to experimentally verify the property.

References

[1] P. Kazanzides, G. Fichtinger, G. Hager, A. Okamura, L. Whitcomb, R. Taylor, Surgical and interventional robotics-core concepts, technology, and design, Robotics & Automation Magazine, IEEE 15 (2008) 122–130.

[2] R. Taylor, A perspective on medical robotics, Proceedings of the IEEE 94 (2006) 1652 –1664.

[3] P. Berkelman, et al., A Compact Modular Teleoperated Robotic System for Laparoscopic Surgery, The International Journal of Robotics Research 28 (2009) 1198.

[4] K. Cleary, C. Nguyen, State of the art in surgical robotics: Clinical applications and technology challenges, Computer Aided Surgery 6 (2001) 312–328.

[5] C. M. R. Marohn, C. E. J. Hanly, Twenty-first century surgery using twenty-first century technology: surgical robotics, Current Surgery 61 (2004) 466–473.

[6] E. Guglielmelli, M. J. Johnson, T. Shibata, Guest editorial special issue on rehabilitation robotics, Robotics, IEEE Transactions on 25 (2009) 477 –480.

[7] J. Desai, N. Ayache, Editorial Special Issue on Medical Robotics, The International Journal of Robotics Research (2009).

[8] T. L. Johnson, Improving automation software dependability: A role for formal methods?, Control Engineering Practice 15 (2007) 1403–1415.

[9] H. Kress-Gazit, Robot challenges: Toward development of verification and synthesis techniques [from the guest editors], Robotics Automation Magazine, IEEE 18 (2011) 22–23.

[10] R. Muradore, D. Bresolin, L. Geretti, P. Fiorini, T. Villa, Robotic Surgery: Formal Verification of Plans, Robotics & Automation Magazine, IEEE 18 (2011) 24–32.

[11] R. Muradore, P. Fiorini, G. Akgun, D. E. Barkana, M. Bonfe, F. Boriero, A. Caprara, G. De Rossi, R. Dodi, O. J. Elle, et al., Development of a cognitive robotic system for simple surgical tasks, Int J Adv Robot Syst 12:37 (2015).

[12] R. Alur, C. Courcoubetis, T. A. Henzinger, P. H. Ho, Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems, in: Hybrid Systems, volume 736 of *LNCS*, Springer, 1993, pp. 209–229.

[13] L. Benvenuti, D. Bresolin, P. Collins, A. Ferrari, L. Geretti, T. Villa, Assume-guarantee verification of nonlinear hybrid systems with ARIADNE, Int. J. Robust. Nonlinear Control 24 (2014) 699–724.

[14] D. Bresolin, L. Geretti, R. Muradore, P. Fiorini, T. Villa, Verification of robotic surgery tasks by reachability analysis: A comparison of tools, in: Digital System Design (DSD), 2014 17th Euromicro Conference on, IEEE, pp. 659–662.

[15] G. Frehse, PHAVer: algorithmic verification of hybrid systems past HyTech, International Journal on Software Tools for Technology Transfer (STTT) 10 (2008) 263–279.

[16] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, O. Maler, SpaceEx: Scalable verification of hybrid systems, in: Proc. 23rd International Conference on Computer Aided Verification (CAV 2011), volume 6806 of *LNCS*, Springer Berlin / Heidelberg, 2011, pp. 379–395.

[17] S. Ratschan, Z. She, Safety verification of hybrid systems by constraint propagation based abstraction refinement, ACM Transactions in Embedded Computing Systems 6 (2007).

[18] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: modelling, planning and control, Springer Verlag, 2008.

[19] R. Alur, T. Dang, F. Ivančić, Counterexample-guided predicate abstraction of hybrid systems, Theoretical Computer Science 354 (2006) 250 – 271.

[20] G. Gössler, S. Graf, M. Majster-Cederbaum, M. Martens, J. Sifakis, An approach to modelling and verification of component based systems, SOFSEM 2007: Theory and Practice of Computer Science (2007) 295–308.

[21] A. Cimatti, S. Tonetta, A property-based proof system for contract-based design, in: Proc. of 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2012), pp. 21–28.