Parametric formal verification: the robotic paint spraying case study

Luca Geretti [*] **Riccardo Muradore** [**] **Davide Bresolin** [***]
Paolo Fiorini [****] **Tiziano Villa** [†]

[*] *University of Verona, Italy (e-mail: luca.geretti@univr.it)*
[**] *University of Verona, Italy (e-mail: riccardo.muradore@univr.it)*
[***] *University of Padova, Italy (e-mail: davide.bresolin@unipd.it)*
[****] *University of Verona, Italy (e-mail: paolo.fiorini@univr.it)*
[†] *University of Verona, Italy (e-mail: tiziano.villa@univr.it)*

Abstract:
The design of robots in industrial automation is based on classical control theory approaches. Recently, formal verification methodologies have been introduced in the design flow, due to their ability of analyzing the model of the robot-environment system in a conservative way. In this paper we specifically explore the analysis of system parameters within a continuous space, by developing an extension of the tool Ariadne for reachability analysis of hybrid automata. Under this framework, the system takes the form of a composition of automata which model discrete control parts that operate in a continuous environment. In particular, the dynamics of the system includes parameters, i.e., unspecified constants for which we want to observe the effect on the dynamics, with the purpose of finding optimal design values. As a case study for this methodology, we consider a robotic paint sprayer, in which we use Ariadne to study the effect of choosing different values of a parameter that represents a point of observation for the system. Using the information gathered from this automated analysis, we provide an answer to the problem of optimizing the surface spraying speed while respecting a given measure of spraying quality.

Keywords: Parametrization, Formal verification, Robotics, Hybrid systems, Nonlinear systems, Reachability, Interval arithmetics

1. INTRODUCTION

According to the Industry 4.0 paradigm, in the factory of the future there will be a tighter integration of cyber-physical systems cooperating at different levels: from sensing to perception, from reasoning to actuation. To fully exploit the potentiality of this new design approach, engineers have to cope with two challenges: (1) develop easy-to-use tools to program the different devices and services in a short time, and (2) check off-line if the designed control architecture will fulfill the requirements and the technical specifications. In this paper we address the second aspect by focusing on formal methods for the industrial plants of the future (Bresolin et al. (2012)). To make the analysis more concrete, we select among the different possible case studies, the robotic painting of a surface through spraying. This is a well known problem studied in particular in the automotive industry, for example in Goodman and Hoppensteradt (1991); Biegelbauer et al. (2005); Chen et al. (2005).

The optimal paint path is usually calculated via manual programming or using computer-aided design (CAD) based on models for the robot manipulator and the shape of the 3D objects, as in Gasparetto et al. (2012); Yu et al. (2015).

Starting from a painting motion for the robotic arms holding the sprayer, formal verification is able to answer questions like the following two:

- is the whole surface well and uniformly covered?
- are there some areas where too much paint has been sprayed?

The first question relates to the quality level of the painting (i.e., uniformity of the paint thickness), whereas the second one refers to an economical issue (i.e., minimum wasted paint).

In this paper in particular we use paint spraying as a case study to focus on the problem of analyzing a system parametrically. By parametric verification we mean the analysis of the effect of varying parameters on the properties of a system, with the purpose of finding the optimal values that satisfy some requirements.

We developed our methodology and applied it on the tool Ariadne, a C++ library that computes reachability analysis with interval arithmetics to perform formal verification in a numerically rigorous way (Collins et al. (2012)). Ariadne uses the well-known hybrid automata model of Alur et al. (1993), which allows to observe the evolution of continuous variables and the switch between different discrete modes of operation. The library has already been applied to robotic problems, for example in

Muradore et al. (2011). In fact, it has been used also to verify systems parametrically: in Benvenuti et al. (2012) for dominance checking of hybrid controllers and in Benvenuti et al. (2014) for assume-guarantee verification. In this paper, we formalize the methodology for parametric verification and extend it in order to identify how to partition the parameter space automatically. The purpose of the proposed approach is to enable spatial analysis of the system in a numerically conservative way, for example to synthesize optimal control parameters with respect to some properties.

Parametric analysis for hybrid automata has already been explored in the current literature, for example in the CORA tool of Althoff (2015) or the KeYmaera tool of Fulton et al. (2015). These projects however focus on linear or affine systems, while ARIADNE is concerned with non-linear behavior.

In the next Section we will describe our methodology and its implementation in ARIADNE. Sec. 3 describes in detail the model for the case study, while in Sec. 4 the verification setup and results are presented. Finally, Sec. 5 draws the conclusions and mentions future developments.

2. PARAMETRIC ANALYSIS

Let us start the discussion by focusing on the evolution of the continuous state, with particular attention to the impact of parameters.

We introduce X as the finite set of n continuous variables representing the continuous state space of the system. Then we call $\boldsymbol{x} \in \mathbb{R}^n$ an assignment of values for the variables in X. Similarly, we consider P as the finite set of m continuous parameters that occur in the model for the system. In this case we assume that the parameters lie within a compact domain D_p. We denote with $\boldsymbol{p} \in D_p$ an assignment of values for the parameters in P.

Now we present the functions within an automaton model where parameters may be used:

(1) Dyn_ℓ: *dynamic* function for each location ℓ, where $\dot{\boldsymbol{x}} = \mathrm{Dyn}_\ell(\boldsymbol{x}, \boldsymbol{p})$ holds;
(2) Inv_ℓ: *invariant* function, which allows evolution within a location ℓ, active when $\mathrm{Inv}_\ell(\boldsymbol{x}, \boldsymbol{p}) \leq 0$; multiple invariant functions for the same location are allowed;
(3) Act_τ: *activation* function for a given transition τ from a location ℓ_1 to a location ℓ_2, active when $\mathrm{Act}_\tau(\boldsymbol{x}, \boldsymbol{p}) \geq 0$;
(4) Res_τ: *reset* function for a given transition τ from a location ℓ_1 to a location ℓ_2, which determines the new value for $\boldsymbol{x}$ on ℓ_2: $\boldsymbol{x} \Rightarrow \mathrm{Res}_\tau(\boldsymbol{x}, \boldsymbol{p})$.

If D_p is a point, i.e., it has an empty interior, the function is *not parametrized* and so it has only one possible behavior, corresponding to a specific choice of the parameters. Instead, if P has a non-empty interior, then the function is *parametrized* with respect to $\boldsymbol{p}$. In this context, we call $f(\boldsymbol{x}, \boldsymbol{p})$ a *singleton instance* of the parametrized function.

The semantics of interval analysis of Moore et al. (2009) establishes that the image of a function over an interval is an overapproximation of the union of the images of the function over all its points. We will use the square-parentheses notation

$$f[\boldsymbol{x}, D_p] \supseteq \bigcup_{\boldsymbol{p} \in D_p} f[\boldsymbol{x}, \boldsymbol{p}] \tag{1}$$

to refer to such inclusion in terms of images.

In addition, let's define $D_p^{(j)}$ as a generic j-th subset of the parametric domain. Now, we want to choose a collection of subsets $\Delta_p = \{D_p^{(j)}\}$ such that $\bigcup_j D_p^{(j)} = D_p$. It holds

$$f[\boldsymbol{x}, D_p] \supseteq \bigcup_{D_p^{(j)}} f\left[\boldsymbol{x}, D_p^{(j)}\right] \supseteq \bigcup_{\boldsymbol{p} \in D_p} f[\boldsymbol{x}, \boldsymbol{p}] \tag{2}$$

where we call $f\left(\boldsymbol{x}, D_p^{(j)}\right)$ a *subset instance* of the parametrized function.

When a parametrized function f is used to model a hybrid system S, we can similarly talk about parametrized systems and their singleton/subset instances. We call F_S the set of parametrized functions in S. It is clear that the aforementioned property on functions translates to the following one: the behavior of a parametrized system includes the union of the behaviors of its instances. Consequently, from Eq. 2 we have:

$$Reach_S(D_p) \supseteq \bigcup_{D_p^{(j)}} Reach_S\left(D_p^{(j)}\right) \supseteq \bigcup_{\boldsymbol{p} \in D_p} Reach_S(\boldsymbol{p}) \tag{3}$$

where $Reach_S$ represents the *reached set* of S from a given initial point $\boldsymbol{x}_I$, up to a given evolution time $t_{\max}$ which depends on the objective of the analysis. Eq. 3 states that we can obtain an overapproximation of $\bigcup_{\boldsymbol{p} \in D_p} Reach_S(\boldsymbol{p})$ by using a finite amount of subset instances, where each instance implies an independent analysis of the parametrized system. Our ultimate goal is to use $Reach_S$ to identify the behavior of the system as a function of $\boldsymbol{p}$ and subsequently verify the given requirements.

2.1 Implementation

Applying Eq. 3, we analyze $Reach_S(\boldsymbol{p}), \forall \boldsymbol{p} \in D_p$ by discretizing the parametric space into Δ_p. Consequently, we should split D_p in the best possible way. For efficiency purposes, we choose the condition

$$D_p^{(j_1)} \cap D_p^{(j_2)} = \emptyset, \forall \left(D_p^{(j_1)}, D_p^{(j_2)}\right) \in \Delta_p \times \Delta_p. \tag{4}$$

How do we choose Δ_p? On one hand, since our analysis time scales linearly with the number of subset instances, we want to minimize the cardinality of Δ_p. On the other hand, from Eq. 1 we recognize that interval arithmetics introduces an overapproximation proportional to the width of the intervals involved; the application to a hybrid system can be regarded as an addition of noise to the reached set. Using very large intervals may therefore produce an unacceptable signal-to-noise ratio in $Reach_S$ with respect to the verification purposes. Additionally, if we want to analyze the system as a function of the parameters' values, we may actually prefer relatively small intervals.

A simple approach would be to split D_p using disjoint intervals of equal user-defined width for each parameter. We obtain Δ_p as the set of all the interval products, which represents a regular tiling of D_p. This solution however

does not account for the impact that different values of a parameter yield over the system behavior.

Therefore, a more adaptive approach requires to identify the effect of varying the size of the subset instance. First, the user provides the minimum desired widths $W_{\min} = \{w_{p_i,\min}\}_{p_i \in P}$ for parameter intervals; the choice of a width value may be determined by the precision of the related parameter to control. Second, given a candidate subset $D_p^{(j)}$, we evaluate

$$\rho^{(j)} = \max_{f \in F_S} \frac{r\left(f\left[D_x^{(f,j)}, D_p^{(j)}\right]\right)}{r\left(f\left[D_x^{(f,j)}, D_{p,\min}^{(j)}\right]\right)} \tag{5}$$

which we call the *spread ratio* of the j-th parameter subset. Here by $r\left(f\left[\cdot\right]\right)$ we refer to the radius of the image of a given function f. In particular, $D_{p,\min}^{(j)}$ refers to the subset having the same center as $D_p^{(j)}$ and widths given by $W_{\min}$. Finally $D_x^{(f,j)}$ is a domain for $\boldsymbol{x}$, calculated for each function and each subset, which represents an approximation of the reached set to be used in a static analysis of the system. There are several possible choices for calculation of this domain, depending on the desired accuracy: currently we use the domain obtained from $Reach_S\left(\boldsymbol{p}_c\right)$, i.e., the reached set of a singleton instance corresponding to the center of D_p, since its computational cost is independent of $|\Delta_p|$.

Summarizing, Eq. 5 is the quadratic mean of ratios between the image radius for a given parameter subset and the image radius of the corresponding subset with minimized widths. It works as a replacement of the gradient with respect to $\boldsymbol{p}$, which is not available in the model. Consequently, the evaluation of $\rho^{(j)}$ allows to determine the local impact of the subset size. While it holds that $\rho^{(j)} \geq 1$ if $D_p^{(j)}$ has widths greater than $W_{\min}$, we want to choose the largest $D_p^{(j)}$ such that

$$\rho^{(j)} < \rho_{\max}. \tag{6}$$

Here $\rho_{\max}$ is a user-defined numerical setting that determines the tolerance for the spread ratio.

Now that we have a measure to decide whether a candidate parameter subset is acceptable, we need a procedure to choose the actual candidates. Our approach relies on a top-down partitioning of D_p according to a multidimensional binary tree. We start by choosing D_p as a candidate and we evaluate it: if it does not satisfy Eq. 6, then we split it along the first parameter and evaluate the resulting two candidates. Each time a candidate is not acceptable, we split along the next parameter; the parameter to split is therefore a function of the tree depth. This procedure continues iteratively in a depth-first fashion, until either we find a suitable candidate or the subset has widths lesser than $W_{\min}$. The result is a partitioning of D_p, such that the largest subsets correspond to the regions where parameters have a smaller impact on the system evolution. The choice of a binary tree representation (which is stored internally using a binary decision diagram) is aimed at efficiency, since it allows the storage of a large amount of subsets with a minimum memory footprint. Other approaches may clearly be envisioned, that explore the parameters space using different cost metrics.

However, it should be noted that when replacing a subset of parameters into F_S, we are forced to decouple the parameters by taking an interval for each parameter. This operation translates into using the *bounding box* of the subset domain: if $\Delta_p = D_p$, the resulting overapproximation may be significant, but for $|\Delta_p| \to \infty$ such effect shrinks to zero. Still, we are unable to guarantee Eq. 4 unless D_p is itself an axis-aligned box.

Finally, given the fact that each splitting halves one dimension of the parameter subset, a reasonable choice for the spread ratio tolerance, under the assumption that F_S is linear in P, is $\rho_{\max} = 2$.

2.2 Configuration

While the methodology above is sound, in our implementation by an automated routine in ARIADNE we allow two user-defined configuration settings: $\rho_{\max}$ and $W_{\min}$. This flexibility is sometimes necessary for fine-tuning the verification, to account for the numerical limitations that the ρ criterion shows with respect to the complexity of the dynamics of the system.

Yet, in certain cases this may still be insufficient, particularly for multi-dimensional parametric spaces coupled with non-linear system dynamics. In those situations, the high overapproximation noise due to system parametrization may prevent the verification altogether.

To overcome such drawback, we offer the ability to use points instead of intervals for one or more parameters. The choice of Δ_p is the same as in the previous Subsection; however, when analyzing the system, we replace the interval for a chosen parameter p_i with its midpoint. Since we are sampling the parametric space, the spatial information gathered by the analysis is limited and sometimes the results are not rigorous. Still, based on the verification problem, it may be sufficient to use intervals only for the most important parameters: this is usually the case when the remaining parameter values are quantized anyway.

3. CASE STUDY MODELING

We model the paint spraying system using three automata, each related to the dynamics of a specific set of variables:

(1) The position $\boldsymbol{x} = (x, y)$ of the sprayer and the velocity v_x;
(2) The exposure s to the spray at a certain observation point $\boldsymbol{x}_o = (x_o, y_o)$;
(3) The paint deposition z at $\boldsymbol{x}_o$.

The corresponding automata are shown in Figs. 1,2,3. All transitions are assumed as urgent, meaning that implicit invariants exist such that the automaton leaves a discrete state as soon as a transition is active.

Before describing each automaton in detail, we can summarize the behavior of the system as follows: given a planar surface, the paint sprayer performs a linear trajectory along x for a given width H, after which it shifts by an amount d along y and reverses its direction for another pass, and so on. The footprint of the spray on the surface

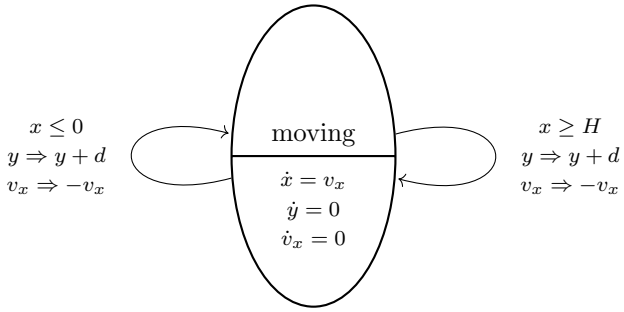

Fig. 1. The automaton for sprayer position $\boldsymbol{x} = (x, y)$ and speed v_x.

is a circle with radius L; however, the spatial deposition of the color in a given instant is nonlinearly proportional to the distance from the center of the circle.

3.1 Automaton for sprayer trajectory

The automaton that models the trajectory of the sprayer spot, shown in Fig. 1, requires one location only: such location is called *moving*, where we set the dynamics for the position $\boldsymbol{x}$ and the speed v_x; the speed v_y is neglected, since we are going to model the y shift as instantaneous for simplicity. This is due to the fact that we are not interested in the behaviour near the borders of the painted surface: for the same reason, we do not model acceleration and deceleration when reversing the velocity. The conditions $x \geq H$ and $x \leq 0$ determine when the transition is taken. The $\Rightarrow$ symbol stands for the *reset* operation, which updates the value of a variable after the transition is taken.

3.2 Automaton for spraying exposure

The automaton for the spraying exposure s, shown in Fig. 2, is a "support" automaton which provides a measure of the incidence of the sprayer over the observed point $\boldsymbol{x}_o = (x_o, y_o)$. The value of s ranges between 0.0 and 1.0, with a maximum when the sprayer is centered at (x_o, y_o), and a minimum as soon as the distance is greater or equal to the sprayer radius L. Therefore, two locations are present: the *far* one, where $||\boldsymbol{x} - \boldsymbol{x}_o|| \geq L$, and the *close* one, where $||\boldsymbol{x} - \boldsymbol{x}_o|| \leq L$. While s remains zero in the *far* location, its expression in the *close* one is modeled as a raised cosine function:

$$s = \frac{1}{2} + \frac{1}{2} \cos \left[\pi \left(\frac{||\boldsymbol{x} - \boldsymbol{x}_o||}{L} \right)^2 \right]. \tag{7}$$

In particular, when modeling in ARIADNE we use the square of the distance since the norm function has numerical issues around zero: the norm function requires a square root, whose domain may partially fall into the negatives when overapproximations are involved.

Since dynamics in ARIADNE are provided in differential form, by calculating the total derivative of s over time we obtain:

$$\dot{s} = -\frac{v_x \pi}{L^2} (x - x_o) \sin \left[\pi \frac{||\boldsymbol{x} - \boldsymbol{x}_o||^2}{L^2} \right]. \tag{8}$$

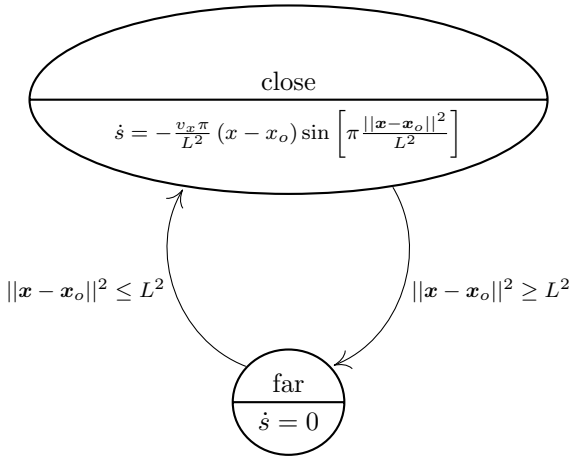

Fig. 2. The automaton for the exposure s to the spray.

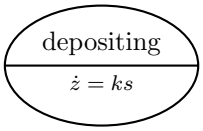

Fig. 3. The automaton for paint deposition z at $\boldsymbol{x}_o$.

3.3 Automaton for paint deposition

The automaton for the behavior of paint deposition z is shown in Fig. 3. Only one location is required, with no transitions: the derivative of z reacts in a proportional way to the exposure, with a constant given by k.

4. VERIFICATION

In this Section we define the objectives of verification, specify the setup and comment on the results that ARIADNE is able to provide.

As anticipated in the Introduction, the objectives are the following:

(1) Limit the spatial deviation in the deposition of paint on the surface;
(2) Keep the deposition of paint below a maximum value.

Combining the two objectives, we identify a lower threshold $z_{\min} = 180\mu m$ and an upper threshold $z_{\max} = 220\mu m$, i.e., we allow a deviation of $\pm 10\%$ around a deposition of $200\mu m$. Consequently, our verification objective becomes:

$$z_{\min} \leq z \leq z_{\max} \tag{9}$$

where we remind here that ARIADNE will always yield a deposition value within an interval.

The parameter set that we choose for the verification is $P = \{d, V, y_o\}$, with

- d: the distance between sprayer passes, introduced to identify how it affects the spatial distribution;
- V: the modulus of the sprayer velocity, introduced to identify how it affects the deposition amount;
- y_o: the y coordinate of the observation point, used to check the spatial distribution of paint deposition in the direction orthogonal to spraying.

The remaining possible parameters are treated as constants, by choosing $L = 1\,cm$, $x_o = L$, $H = 2L$, $k = 0.001$. Given $X = \{s, v_x, x, y, z\}$, the initial set is $\{0, V, 0, 0, 0\}$ and the exposure automaton starts from the far location.

The domain for the parameters is

$$D_p = \{[L/4, L], [5\,cm/s, 40\,cm/s], [4d, 4d + d/2]\}. \quad (10)$$

In particular, we chose a domain width for y_o equal to $d/2$ due to the apparent y symmetry of the paint deposition in respect to the spray pass line. The maximum value for d equal to L is reasonable: in other terms, we want at least two passes to affect the region between two lines; the minimum value instead implies that four passes contribute to the total deposition on any point in such region. Additionally, the choice of a minimum y_o value of $4d$ guarantees that all the passes that would affect D_{y_o} are always performed. This implicitly means that after we have covered a distance of $8d$ along y, we do not need to evolve the system further: this represents our termination condition for the system, which otherwise would evolve indefinitely. Also notice that we put the observation point x_o in the center of the spray pass line, and chose H such that the line is as short as necessary in respect to L.

Summarizing, we made some choices regarding constant values and parametric domain with the purpose of reducing the (X, P) space of the system to the minimum required, thus minimizing the verification time. On the contrary, if for example we decided for an axis-aligned box for D_p, we would have considered the worst case scenario for y_o and used the $[L, 2L]$ interval instead. Apart from having a larger interval width, the resulting domain also covers regions of the parametric space in which we are not actually interested: it can be shown that the volume of D_p is more than twice compared to our solution.

Now that we described completely the system model, we can notice a peculiarity: the parameter V actually does not appear anywhere within the functions, but it enters as an initial value. From Fig. 1, V affects indirectly the value of v_x. According to Eq. 5, the impact of V comes from $D_x^{(f,j)}$ rather than $D_p^{(j)}$. Consequently, we can see that a parameter is not required to appear in any f directly, as long as it influences $D_x^{(f,j)}$. For comparison, y_0 influences directly the dynamics and guards of Fig. 2, while d appears in the resets of Fig. 1.

As discussed in Sec. 2, since we are allowed to use points instead of intervals for some parameters, here, we adopt this simplification for d and V. The motivation is that these two parameters are not critical, while on the other hand y_o must be analyzed continuously to identify the spatial range of values. Finally, we choose $W_{\min} = \{0.4\,mm, 20\,mm/s, 50\,\mu m\}$ and $\rho_{\max} = 2$.

4.1 Results

First, Fig. 4 details the spatial behavior of the deposition over y_o for some values of V, in particular the minimum, maximum and central points of the velocity domain; we chose $d = L$ for these samples. First we notice that z is expressed as an interval, which takes into account the overapproximation noise. The fact that the noise is larger towards the right (i.e., towards the midpoint between two passes) is due to the increasing superposition of the passes, while on the leftmost point only one pass contributes to the deposition. We can also see that the relation between z and V is non-linear. Finally, even if not apparent from the figure, the curves have the same shape regardless of the velocity.

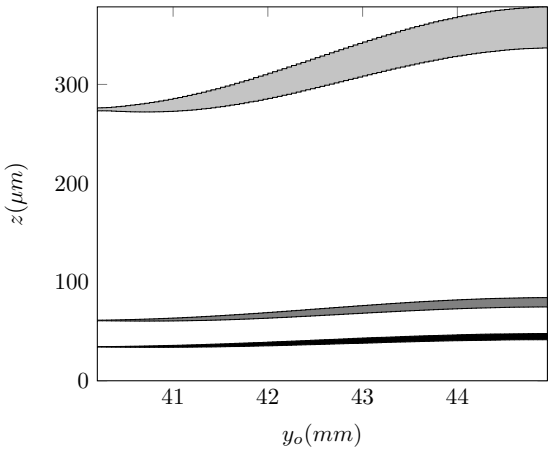

Fig. 4. The z curves for $d = L$ and different values of V: $\{5cm/s, 22.5cm/s, 40cm/s\}$, from top to bottom.

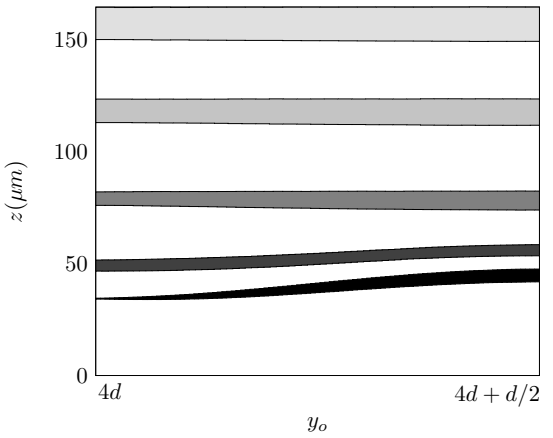

Fig. 5. The z curves for $V = 40\,cm/s$ and different values of d: $\{L/4, L/3, L/2, 3L/4, L\}$, from top to bottom.

Fig. 5 instead shows how the deposition varies along y_o for different values of d. Please note that we spread the results along a common $[4d, 4d + d/2]$ domain for y_o. From the $\{L/4, L/3, L/2, L\}$ subset, we can see that the deposition is linear with respect to the number of passes that affect a given y_o. However, the more passes interact, the larger the numerical issues: in practice, the overapproximation noise cancels information on the curve and limits observability to the minimum/maximum values. The width of such noise clearly increases as d decreases.

Finally, in Fig. 6 we summarize how the deposition satisfies Eq. 9 while choosing subsets for all the parameters according to D_p and $W_{\min}$. We take the convex hull of z for all the y_o results, in order to project on the $d - V$ subspace. While for d and V we analyze by points rather than by intervals, for graphical reasons we plot the results within boxes. A grey box implies that only the midpoint of z lies within the required bounds, while a black box implies Eq. 9 is satisfied for all z.

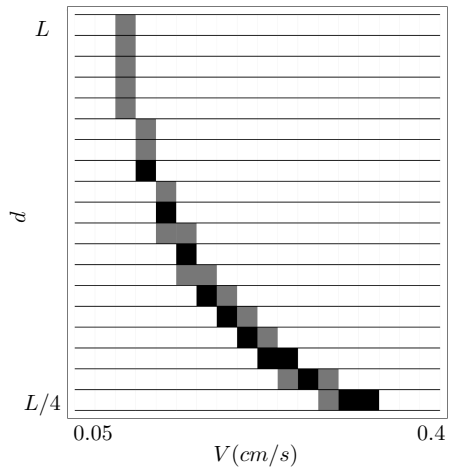

Fig. 6. Verification results for all the d-V samples: grey boxes imply that the average z is within the bounds, black boxes that all z values are within the bounds.

It must be remarked that finer $W_{\min}$ values would have yielded a more accurate curve, but still we can infer that the behavior is near-hyperbolic, where on the lower-left region we have an excessive deposition, and on the upper-right region an insufficient deposition. However, lower values of d turn out to yield a larger range of valid V values. We also calculated the optimal d-V point for painting speed, expressed by the $d \times V$ product, as $(d = 0.25, V = 0.33)$.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a sound methodology for performing parametric formal verification of a system. The paint spraying case study showed that it is possible to extract optimal design values in a numerically conservative way, such that the desired system properties are guaranteed with respect to the provided model. Future work on this specific topic will focus on studying different partitioning techniques, and on exploiting more sophisticated spread ratio measures.

In terms of numerical issues, we recognize that computing reachable sets for multiple parameters introduces a significant overapproximation noise, which must be dealt with. There exist several ways to increase the capabilities of ARIADNE to perform analysis for multidimensional parameter spaces. Currently, we are working to improve the continuous evolution routines to better control the error. Moreover, a significant advantage would be the ability to analyze each automaton in isolation, by substituting an input variable with a *differential inclusion*, as in Zivanovic and Collins (2010). This enhancement in general would enable *contract-based design* (see Nuzzo et al. (2015)), which simplifies the translation of design requirements into formal verification procedures.

REFERENCES

Althoff, M. (2015). An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems, 2015*.

Alur, R., Courcoubetis, C., Henzinger, T.A., and Ho, P.H. (1993). Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, volume 736 of *LNCS*, 209–229. Springer, Lyngby, Denmark.

Benvenuti, L., Bresolin, D., Collins, P., Ferrari, A., Geretti, L., and Villa, T. (2012). Ariadne: Dominance checking of nonlinear hybrid automata using reachability analysis. In *Proc. of the 6th International Workshop on Reachability Problems (RP12)*, volume 7550 of *LNCS*, 79–91. Springer Berlin Heidelberg.

Benvenuti, L., Bresolin, D., Collins, P., Ferrari, A., Geretti, L., and Villa, T. (2014). Assume-guarantee verification of nonlinear hybrid systems with ARIADNE. *Int. J. Robust. Nonlinear Control*, 24(4), 699–724.

Biegelbauer, G., Pichler, A., Vincze, M., Nielsen, C.L., Andersen, H.J., and Haeusler, K. (2005). The inverse approach of flexpaint [robotic spray painting]. *IEEE Robotics Automation Magazine*, 12(3), 24–34.

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

Chen, H., Xi, N., Sheng, W., Dahl, J., and Chen, H. (2005). Analysis of system performance for robotic spray forming process. In *2005 IEEE/RSJ Intern. Conference on Intelligent Robots and Systems*, 1396–1401.

Collins, P., Bresolin, D., Geretti, L., and Villa, T. (2012). Computing the evolution of hybrid systems using rigorous function calculus. In *Proc. of the 4th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS12)*, 284–290. Eindhoven, The Netherlands.

Fulton, N., Mitsch, S., Quesel, J., Völp, M., and Platzer, A. (2015). KeYmaera X: An aXiomatic tactical theorem prover for hybrid systems. In *Proceedings of the International Conference on Automated Deduction, CADE'15, Berlin, Germany*, volume 9195, 527–538. Springer.

Gasparetto, A., Vidoni, R., Pillan, D., and Saccavini, E. (2012). Automatic path and trajectory planning for robotic spray painting. In *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, 1–6.

Goodman, E.D. and Hoppensteradt, L.T.W. (1991). A method for accurate simulation of robotic spray application using empirical parameterization. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, 1357–1368 vol.2.

Moore, R.E., Kearfoot, R.B., and Cloud, M.J. (2009). *Introduction to Interval Analysis*. SIAM.

Muradore, R., Bresolin, D., Geretti, L., Fiorini, P., and Villa, T. (2011). Robotic surgery. *IEEE Robotics Automation Magazine*, 18(3), 24–32.

Nuzzo, P., Sangiovanni-Vincentelli, A.L., Bresolin, D., Geretti, L., and Villa, T. (2015). A platform-based design methodology with contracts and related tools for the design of cyber-physical systems. *Proceedings of the IEEE*, 103(11), 2104–2132.

Yu, Q., Wang, G., and Chen, K. (2015). A robotic spraying path generation algorithm for free-form surface based on constant coating overlapping width. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE Intern. Conf. on*, 1045–1049.

Zivanovic, S. and Collins, P. (2010). Numerical solutions to noisy systems. In *49th IEEE Conference on Decision and Control (CDC)*, 798–803.