UTOPIC: Under-Approximation Through Optimal Control

Josu Doncel[1], Nicolas Gast[2], Mirco Tribastone[3]
Max Tschaikowski[4], and Andrea Vandin[5]

[1] University of the Basque Country, Spain
[2] Inria, Univ. Grenoble Alpes, France
[3] IMT School for Advanced Studies Lucca, Italy
[4] Technische Universität Wien, Austria
[5] DTU Compute Lyngby, Denmark

Abstract. We consider a class of nonlinear systems of differential equations with uncertainties, i.e., with lack of knowledge in some of the parameters that is represented by a time-varying unknown bounded functions. An under-approximation of such systems consists of a subset of its reachable set, for any value of the unknown parameters. By relying on optimal control theory through Pontryagin's principle, we provide an algorithm for the under-approximation of a linear combination of the state variables in terms of a fully automated tool-chain named UTOPIC. This allows to establish tight under-approximations of common benchmarks models with dimensions as large as sixty-five.

Keywords: Under-Approximation, Uncertain Nonlinear Dynamics, Modeling

1 Introduction

Many safety-critical systems can be modeled as nonlinear ordinary differential equations (ODEs) which exhibit uncertain parameters due to finite precision measurements, lack of data, or noise [5]. In order to formally decide whether such a system is safe, it is necessary to determine whether the model can reach a bad state [14, 9, 36]. Since closed-form expressions for reachable sets of nonlinear ODE systems are not known in general [35], it becomes necessary to over- and under-approximate the reachable set. Over-approximation provides a superset of the reachable set, and thus can be used to prove that a model under study is safe. Instead, under-approximation gives a subset of the reachable set; therefore, it can be used to prove the presence of problems. While conceptually different, over- and under-approximation techniques are technically closely related.

In the case of linear dynamics with uncertain parameters, the reachable set can be shown to be convex. This has paved the way for efficient over- and under-approximation techniques for systems of linear ODEs, see for instance [34, 28, 27, 4] and references therein. Under-approximation techniques for nonlinear ODEs received less attention in the past, and often rely on over-approximation

techniques for nonlinear systems. Indeed, by relying on the Hamilton-Jacobi equation, a special case of the Hamilton-Jacobi-Bellman equation used in over-approximation techniques [33], the recent work [38, 51] constructs a sequence of convex programs for polynomial ODE systems. The sequence is proven to converge to the actual reachable set but is truncated in practical computations to ensure a compromise between tightness and computational cost. [51] has been extended to cover time-varying uncertainties and non-singleton initial sets [52]. In [16, 53], instead, the main idea is to reverse time of the original ODE system and to over-approximate the so obtained backward flow while ensuring certain topological properties of the over-approximation. Similarly to [51], the work [53] was extended to cover time-varying uncertainties and non-singleton initial sets [54]. Another line of research is [29] and its extension [30] where over- and under-approximations are obtained via Taylor models.

The aforementioned approaches share the following features:

(i) the full reachable set (and not some projection of it) is approximated;
(ii) the approximation is given across the whole time course;
(iii) in addition to ODE parameters, the initial conditions may or must be uncertain.

Instead, the technique (presented in Section 2) and the tool (presented in Section 3) — UTOPIC — presented in this paper focus on the under-approximation of a one-dimensional projection of the reachable set at a given time point in the presence of time-varying uncertainties and fixed initial condition. The one-dimensional projection is given by a linear combination of state variables. Our restriction essentially amounts to computing an under-approximation of a one-dimensional linear projection of the reachable set; the whole time course can then be covered by performing the under-approximation for each grid point of a sufficiently fine discretization of the time interval.

The basic idea underlying our approach is to interpret uncertain parameters as controls and to minimize and maximize a given linear combination of state variables using Pontryagin's maximum principle, a well-established technique from optimal control theory [31]. The interval spanned by the so-obtained extrema can be shown to be an under-approximation of the reachable set of the linear combination. Moreover, thanks to the fact that Pontryagin's maximum principle is a necessary condition for optimality, the aforementioned interval is likely to be tight enough to cover the actual reachable set in many cases. This is confirmed by a numerical evaluation of UTOPIC in Section 4 on classical benchmark models from [15, 24], where the under-approximation is compared against over-approximations computed by state-of-the-art reachability analysis tools such as Flow* [15] and CORA [3]. Further, we show the scalability of the algorithm on larger scale models from biochemistry [18, 13] with 65 state variables.

Related Work. The most closely related approaches are [7, 10] which propose the use of Pontryagin's maximum principle, but for over-approximation. Due to the fact that Pontryagin's maximum principle is only a necessary condition for

optimality such a solution provides only a heuristic estimation which is not guaranteed to be valid in general [10]. The authors of [7] address this by restricting to bilinear systems for which Pontryagin's maximum principle can sometimes be shown to be a sufficient condition for optimality.

We next provide a brief account on over-approximation techniques for nonlinear ODEs. The abstraction approach locally approximates the nonlinear dynamics by multivariate polynomials or affine maps, see [6, 18, 15, 2, 16]. While abstraction techniques can cover many practical models, they are prone to the curse of dimensionality, i.e., their run-time may be exponential in the size of the ODE system, as discussed in [2, 22]. A similar remark applies to the worst-case complexities of techniques relying on the Hamilton-Jacobi-Bellman equation [33] and Satisfiability Modulo Theories (SMT) solvers [32], even though certain combinations with model reductions exist [47]. Another classic approach is based on Lyapunov-like functions known from the stability theory of ODEs [44, 55, 21, 48]. Unfortunately, for nonlinear systems the automatic computation of Lyapunov-like functions remains a challenging task. Restricting to special classes of such functions (e.g., sum-of-squares polynomials [26]) leads to efficient construction algorithms which may provide tight bounds, but existence is not guaranteed. Approximations with differential inequalities [45, 49] and interval arithmetics [46, 40], on the other hand, can be computed efficiently, but are loose in general [10].

To the best of our knowledge, C2E2 [24] is the only tool that supports under-approximation, but only for linear systems with time-varying uncertain parameters [22]. Instead, over-approximation is supported for nonlinear systems with time-invariant uncertain parameters [23]. As UTOPIC covers the under-approximation of nonlinear systems with time-varying uncertain parameters (and C2E2 relies on closed-form expressions in the case of linear systems that will outperform any known over- and under-approximation technique for nonlinear systems), Flow* and CORA are the most related tools to our work because they both support nonlinear systems with time-varying uncertain parameters. That is why we chose them for our numerical tests. In a broader context, we mention here also the tools Breach [20] and S-Taliro [42] which can be used to falsify given properties by relying on approximate sensitivity analysis and randomized testing, respectively.

2 Theoretical Background

Problem statement. We consider a system of ODEs $\dot{x} = f(x, u)$ over state variables $x_1, \ldots, x_n$, and parameter variables $u_1, \ldots, u_m$. We assume that our dynamical system is affine in controls [39], i.e., it holds that

$$f(x, u) = g_0(x) + \sum_{j=1}^{m} u_j g_j(x),$$

for some differentiable functions $g_0, g_1, \ldots, g_m : \mathbb{R}^n \to \mathbb{R}^n$.

Given an initial condition $x(0) \in \mathbb{R}^n$, a finite time horizon $T > 0$, and a weight vector $\alpha{=}(\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n$, our goal is to compute an under-approximation at time T of the one-dimensional projection

$$\mathcal{P}_\alpha(T) = \Big\{ \sum_{i=1}^{n} \alpha_i x_i(T) \mid \dot{x} = f(x,u), u \in \mathcal{U}(\underline{\beta}, \overline{\beta}) \Big\},$$

where the set $\mathcal{U}(\underline{\beta}, \overline{\beta})$, defined as

$$\mathcal{U}(\underline{\beta}, \overline{\beta}) = \{u : [0;T] \to \prod_{j=1}^{m} [\underline{\beta}_j; \overline{\beta}_j] \mid u \text{ measurable}\},$$

describes the admissible parameter functions. We assume that $\dot{x} = f(x,u)$ has a solution on $[0;T]$ for any admissible function u, thus excluding in particular finite explosion times.

We seek to compute parameter functions $\underline{u}$, $\overline{u} \in \mathcal{U}(\underline{\beta}, \overline{\beta})$ such that

$$\begin{aligned} \underline{u} &\in \arg\min \Big\{ \sum_{i=1}^{n} \alpha_i x_i(T) \mid u \in \mathcal{U}(\underline{\beta}, \overline{\beta}), \dot{x} = f(x,u) \Big\} \\ \overline{u} &\in \arg\max \Big\{ \sum_{i=1}^{n} \alpha_i x_i(T) \mid u \in \mathcal{U}(\underline{\beta}, \overline{\beta}), \dot{x} = f(x,u) \Big\} \end{aligned} \tag{1}$$

Thanks to the fact that our dynamical system is affine in controls, the above optimization problems have solutions [39].

Problem solution. We tackle the optimization problems (1) by observing that Pontryagin's principle implies that any optimal parameter function $\underline{u}$ (resp. $\overline{u}$) solves, almost everywhere on $[0;T]$, the differential inclusion

$$\dot{x}(t) = f(x(t), u(t)) \tag{2}$$

$$\dot{p}(t) = -(\partial_x H)(x(t), u(t), p(t)) \tag{3}$$

$$u(t) \in \arg\max_{\underline{\beta} \le u \le \overline{\beta}} H(x(t), u(t), p(t)) \tag{4}$$

where $H(x,u,p) = \langle p, f(x,u) \rangle$ denotes the Hamiltonian, $\langle \cdot, \cdot \rangle$ refers to the scalar product on $\mathbb{R}^n$, and we set $p_i(T) = -\alpha_i$ (resp., $p_i(T) = \alpha_i$) for all $1 \le i \le n$ in the case of minimization (resp., maximization); in (4), the notation $\underline{\beta} \le u \le \overline{\beta}$ is shorthand for $\underline{\beta}_j \le u_j \le \overline{\beta}_j$ for all $1 \le j \le m$.

It is possible to approximate an optimal control satisfying (2-4) by relying on a gradient projection algorithm [31, Section 6.2]. To this end, we consider the cost function

$$J : \mathcal{U}(\underline{\beta}, \overline{\beta}) \to \mathbb{R}, \quad u \mapsto \sum_{i=1}^{n} \alpha_i x_i(T),$$

where x is a solution of $\dot{x}(t) = f(x(t), u(t))$. The main idea to lift the common gradient descent algorithm over reals to the space of functions. More specifically, one interprets J as a functional on the Hilbert space $\mathcal{H} = L^\infty([0;T])$, where

$$L^\infty([0;T]) = \{h : [0;T] \to \mathbb{R}^m \mid h \text{ is measurable and } \|h(t)\|_\infty \leq C \text{ for almost all } t \text{ for some finite } C > 0\}$$

endowed with the scalar product $\langle g, h\rangle = \int_0^T g(s)h(s)ds$. With this, it is possible to prove (see [43]) that J is Fréchet differentiable at any function vector $u \in \mathcal{U}(\underline{\beta}, \overline{\beta}) \subseteq \mathcal{H}$ and that its derivative is given by the function vector

$$(\partial J)(u) : t \mapsto -(\partial_u H)(x(t), u(t), p(t))$$

That is, the value of the derivative at the function vector $t \mapsto u(t)$ is given by the function vector $t \mapsto -(\partial_u H)(x(t), u(t), p(t))$. (Note the similarity to the common gradient descent method over the reals where the value of the derivative at a point vector is given by another point vector.)

In order to compute $(\partial J)(u)$ for a given u, we first solve (2); afterwards, we compute p from (3) which depends on u and x. Similarly to the gradient projection algorithm in finite dimensions, a given $u \in \mathcal{U}(\underline{\beta}, \overline{\beta})$ is either stationary or there is some $\gamma > 0$ such that

$$J\big(\mathfrak{P}\big(u + \gamma \cdot (\partial J)(u)\big)\big) < J(u),$$

where $\mathfrak{P} : \mathcal{H} \to \mathcal{U}(\underline{\beta}, \overline{\beta})$ is the projection from $\mathcal{H}$ onto $\mathcal{U}(\underline{\beta}, \overline{\beta})$. Following [43] and references therein, $\mathfrak{P}(v) \in \mathcal{U}(\underline{\beta}, \overline{\beta})$, where $v \in \mathcal{H}$, is given by

$$(\mathfrak{P}(v))(t) = \begin{cases} v_i(t) & \text{if } \underline{\beta}_i \leq v_i(t) \leq \overline{\beta}_i \\ \overline{\beta}_i & \text{if } v_i(t) > \overline{\beta}_i \\ \underline{\beta}_i & \text{if } v_i(t) < \underline{\beta}_i \end{cases}$$

We are now in a position to perform a gradient (descent) projection algorithm that computes an approximation of an optimal control u: Algorithm 1 takes as inputs the maximal number of steps ν, the positive step sizes $\gamma^1, \gamma^2, \ldots, \gamma^\nu$ and the numerical threshold $\varepsilon \geq 0$ which triggers a termination based on a relative convergence criterion when $|J(u^k) - J(u^{k-1})| < \varepsilon$, where u^k denotes the approximation of the optimal control u obtained after k steps.

The following result can be proven.

Theorem 1. *For a sufficiently small step $\gamma_0 > 0$, Algorithm 1 induces, if applied to $\nu = \infty$, $\varepsilon = 0$ and the constant sequence $(\gamma_0, \gamma_0, \ldots)$, a sequence of controls $(u^k)_k$ which, in turn, induces a decreasing sequence $(J(u^k))_k$ which converges to a stationary control of the cost functional J.*

Proof. In the case of unbounded controls, the proof is at the heart of the calculus of variations, see [39, Section 3.4]. The case of bounded controls, instead, requires a refined reasoning, see [39, Chapter 5] and [43]. □

```
Gradient(ẋ = f(x,u), x(0), α, β, γ, T, ε, ν) :=
  set k = 1
  choose some initial parameter function u^k
  while(true)
    solve ẋ^k = f(x^k, u^k) forward in time using u^k and x(0)
    solve ṗ^k = −(∂_x H)(x^k, u^k, p^k) backward in time using u^k, x^k, p(T)
    compute J^k = Σ_{i=1}^n α_i x^k(T)
    compute u^{k+1}(·) = 𝔓(u^k(·) − γ^k · (∂_u H)(x^k(·), u^k(·), p^k(·)))
    if(k ≥ ν ∨ (k ≥ 2 ∧ |J^k − J^{k−1}| < ε))
      break
    else
      set k = k + 1
    end
  end
  return u^{k+1}
```

Algorithm 1: Gradient projection algorithm for solving (1). Inputs are: the ODE system $\dot{x} = f(x,u)$, the initial values $x(0)$, the weight vector α, the boundary vector β, the time horizon T, a sequence of positive step sizes γ, the numerical threshold $\varepsilon \geq 0$, and the maximal number of steps $\nu \geq 0$.

We remark that Algorithm 1 requires the user to provide a sequence of step sizes $(\gamma^k)_k$. Those are not known a priori and have to be either inferred through trial and error or estimated by some other means. In the case of unconstrained optimization, estimations can be obtained via backtracking. For constrained optimization such as here, instead, the basic line search algorithm can be used. For more information on step size estimation, see [19]. Another point worth noticing is that $(J(u^k))_k$ may converge to a saddle, instead of an extreme, control. In practice, this problem can be tackled by introducing a random noise term. A formal proof of this heuristics has been recently established in the finite dimensional case [37].

We invoke Algorithm 1 for the computation of optimal control candidates $\underline{u}'$ and $\overline{u}'$ for $\underline{u}$ and $\overline{u}$, respectively, see (1). As stated next, $\underline{u}'$ and $\overline{u}'$ induce solution points $\underline{x}'(T)$ and $\overline{x}'(T)$ whose projections span a subinterval of $\mathcal{P}_\alpha(T) \subseteq \mathbb{R}$.

Theorem 2. *Assume that $\underline{u}'$ and $\overline{u}'$ are candidates for $\underline{u}$ and $\overline{u}$ underlying $p(T) = -\alpha$ and $p(T) = \alpha$, respectively. Then, if $\underline{x}'$ and $\overline{x}'$ denotes the solution underlying $\underline{u}'$ and $\overline{u}'$, respectively, it holds that*

$$\big[\sum_{i=1}^{n} \alpha_i \underline{x}'_i(T); \sum_{i=1}^{n} \alpha_i \overline{x}'_i(T)\big] \subseteq \mathcal{P}_\alpha(T)$$

when $\sum_{i=1}^n \alpha_i \underline{x}'_i(T) \leq \sum_{i=1}^n \alpha_i \overline{x}'_i(T)$.

Proof. Given two arbitrary piecewise continuous functions $\underline{u}'$ and $\overline{u}'$, we define the function $u'_\lambda = \lambda \underline{u}' + (1-\lambda)\overline{u}'$ for all $0 \leq \lambda \leq 1$ and note that $u'_\lambda \in \mathcal{U}(\underline{\beta}, \overline{\beta})$

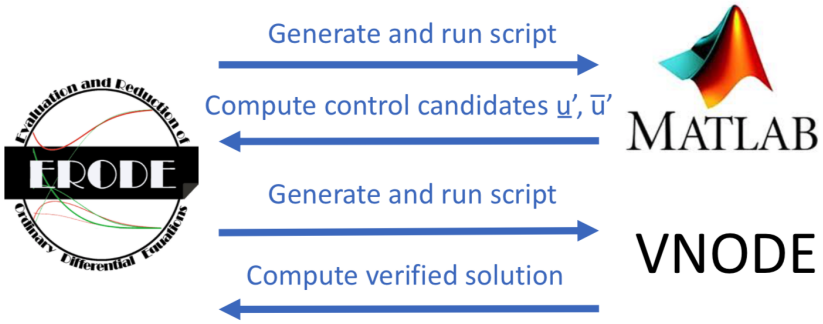

Fig. 1: Pictorial representation of UTOPIC's workflow.

for all $0 \leq \lambda \leq 1$. Using this, we further define $\Phi(\lambda) = \sum_{i=1}^{n} \alpha_i x_i(T)$, where x satisfies $\dot{x} = f(x, u'_\lambda)$. Thanks to the intermediate value theorem, it thus suffices to show that Φ is continuous. Since for any $C > 0$ and $\varepsilon > 0$ there exists a $\delta > 0$ such that $|\lambda - \lambda'| < \delta$ implies $\|f(x, u'_\lambda(t)) - f(x, u'_{\lambda'}(t))\|_\infty \leq \varepsilon$ for all $x \in [-C; C]^n$ and $0 \leq t \leq T$, Gronwall's inequality yields the claim. □

3 UTOPIC

Architecture. Our approach is implemented as a fully automated workflow named UTOPIC—under-approximation through optimal control. UTOPIC involves the following tools: ERODE, MATLAB, and VNODE [41], where:

- ERODE [12], a Java-based cross-platform tool for the specification, evaluation, and reduction of nonlinear ODEs, providing an integrated development environment.
- MATLAB, and in particular the *Symbolic Math Toolbox* to symbolic computation.
- VNODE [41], a C++-based validated solver for computing bounds on the solution of ODEs. Traditional ODE solvers compute approximate solutions. Instead, VNODE first proves that a unique solution exists, and then computes formal bounds that are guaranteed to contain it.

Workflow. UTOPIC is implemented within the ERODE environment by means of the dedicated `utopic` command. The input is a specification of a nonlinear ODE system. Currently, this consists of the differentiable fragment of the IDOL language [11], which corresponds to nonlinear ODEs with derivatives that are multivariate rational expressions over the system's variables. The execution of the ERODE specification triggers a fully automated orchestration, as depicted in Figure 1. In particular ERODE performs the following actions.

1) Generate and execute a MATLAB specification implementing Algorithm 1 for the ODE under study, in order to compute the control candidates $\underline{u}'$, and $\overline{u}'$. This activity is delegated to an external tool to leverage an already available symbolic engine in order to perform the differentiations in (2)–(4) . We remark that MATLAB is the tool of choice in this version of UTOPIC, although it is in principle possible to allow for other engines such SymPy for Python.
2) Given the control candidates $\underline{u}'$, and $\overline{u}'$, create, compile, execute a C++ source file using the VNODE library in order to compute a *verified* solution of the ODE system $\dot{x} = f(x, u)$. This step is required for obtaining a formally guaranteed under-approximation. It is in principle possible to allow for quicker heuristic experimentation by using ERODE's built-in ODE numerical integrators.
3) Present the VNODE-computed under-approximation to the user within the ERODE environment.

Deployment. ERODE does not require any installation process, and it is available, together with a manual, at http://sysma.imtlucca.it/tools/erode. Instead, http://www.cas.mcmaster.ca/~nedialk/ provides installation notes on VNODE.

Example. Figure 2 shows a screenshot of the standard Brusselator model [15, 24] defined in the ERODE editor, together with an example of the `utopic` command which triggers the analysis workflow. In particular, the file defines the ODEs

$$\begin{aligned} \dot{x} &= x^2 y - bx + a \\ \dot{y} &= -x^2 y + cx \end{aligned} \tag{5}$$

with ODE variables x, y and parameters a, b and c. The initial conditions are chosen to be $x(0) = 1.0$ and $y(0) = 1.2$. Instead, the uncertain parameters u are specified using the `paramsToPerturb` argument, while the weight vector α specifying the query is given with the `coefficients` argument. Finally, `delta` specifies the numerical error threshold ε of Algorithm 1, while `step` is used to describe a sequence of uniform step sizes $\gamma = (\texttt{step}, \texttt{step}, \ldots)$. The maximal number of iterations performed by the algorithm is given by `kMax`. Instead, `integrationStep` specifies the integration time step used in the algorithm.

In the case of the Brusselator model, setting `coefficient = { x : 1.0 }` and `tEnd = 1.0` ensures that `utopic` computes an under-approximation of the reachable set

$$\Big\{ 1.0 \cdot x(1.0) \;\Big|\; (\dot{x}(t), \dot{y}(t)) = f(x(t), y(t), a(t), b(t), c(t)) \\ \text{where } a(\cdot) \in [0.9; 1.1], b(\cdot) \in [2.3; 2.7], c(\cdot) \in [1.1; 1.6] \text{ and } t \in [0; 1.0] \Big\},$$

where f is as in (5). Note that `paramsToPerturb` overwrites the parameter choice $a = 1.0$, $b = 2.5$ and $c = 1.5$ at the beginning of the file; we require the specification of all parameters because only a subset of parameters is subject to

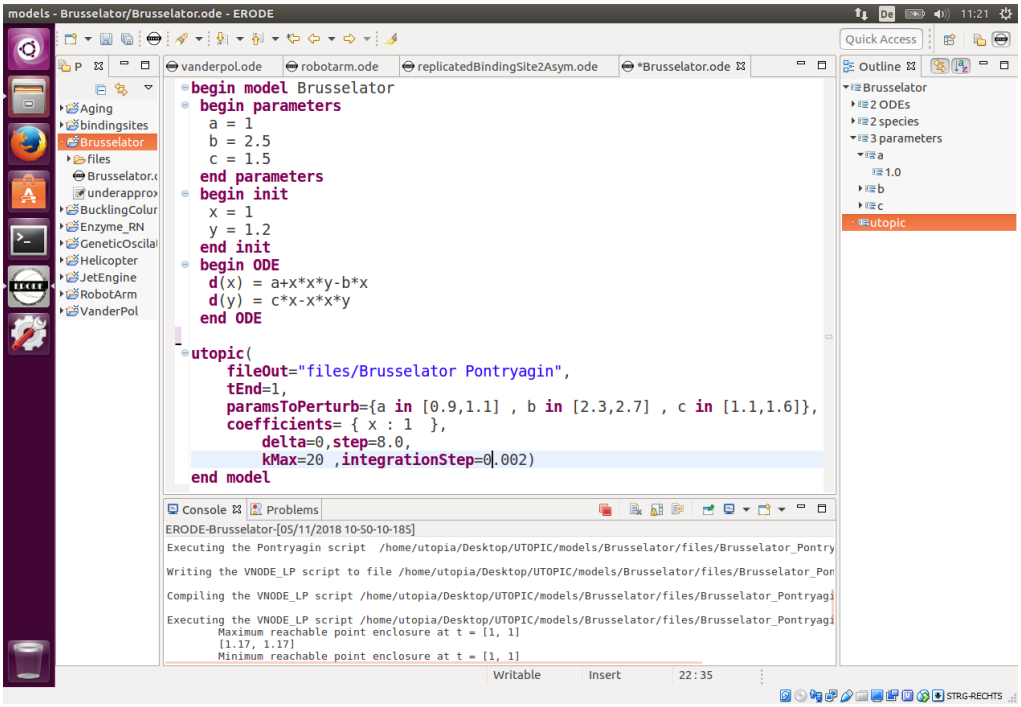

Fig. 2: Graphical-user interface of ERODE in the case of the standard Brusselator model [15, 24]. The ERODE file defines an ODE system with two ODE variables (x and y) and three parameters (a, b and c). In the visualized example, all three parameters are subject to uncertainty, thus turning them in time-varying uncertain parameter functions. For instance, $a(t) \in [0.9; 1.1]$ for all $t \in [0; 1.0]$.

uncertainty in general. The field `coefficient` can be used to describe arbitrary weighted sums of state variables, e.g., `coefficient = { x : 0.5, y : 2.0 }` leads to the under-approximation of the set

$$\Big\{0.5 \cdot x(1.0) + 2.0 \cdot y(1.0) \;\Big|\; (\dot{x}(t), \dot{y}(t)) = f(x(t), y(t), a(t), b(t), c(t)) \\ \text{where } a(\cdot) \in [0.9; 1.1], b(\cdot) \in [2.3; 2.7], c(\cdot) \in [1.1; 1.6] \text{ and } t \in [0; 1.0]\Big\}.$$

Executing the model in Fig. 2 presents the under-approximation results as

```
Maximum reachable point enclosure at t = [1.0, 1.0]
[1.17, 1.17]
Minimum reachable point enclosure at t = [1.0, 1.0]
[0.578, 0.578]
```

This is the processed output of VNODE execution. Being formally an over-approximation technique that is based on interval arithmetics [46, 40], VNODE

returns intervals instead of points. This is because any numerical integration scheme introduces truncations due to finite-precision arithmetics. Thus, already after one time step, the solution of a verified ODE solver has to be represented by a (very small) set instead of a point. Algorithmically, verified integration therefore corresponds to over-approximation in the special case where the initial set is a singleton. This greatly simplifies the task and often leads to over-approximation sets with negligibly small diameters. In the above example, for instance, VNODE returns $[1.17, 1.17]$ and $[0.578, 0.578]$.

Complexity analysis. The complexity of Algorithm 1 depends on the number of state and parameter variables n and m, respectively, the step sizes $(\gamma_1, \gamma_2, \ldots)$, the numerical threshold ε and the maximal number of iterations ν. Assuming that ν is finite, the approach requires one to solve a sequence of n dimensional ODE systems whose length is polynomial in ν and m. Heuristic ODE solvers like `ode45` of MATLAB enjoy a polynomial time and space complexity in n and m. Instead, the complexity of the two VNODE invocations in the last step of the workflow, see Fig. 1, may be computationally expensive and depend, in general, on factors such as the stiffness of the ODE system, the time discretization step and the order of the integration scheme. This notwithstanding we stress that many practical models can be covered efficiently because the initial set is a singleton, as outlined in the preceding paragraph.

4 Evaluation

Set-up. In this section we evaluate UTOPIC on benchmark models from the literature. The main measure of performance is the scalability with respect to the ODE system size. In addition, we investigated the tightness of the under-approximation by comparing it against an over-approximation, since the exact reachable sets are not known for these models. For this comparison we used the tools Flow* [15] and CORA [3]. Ariadne [8] or HySAT/iSAT [25] were not considered because they have been compared to Flow* already [15]; similarly, C2E2 [24] was not considered because it does not support time-varying uncertainties.

In all experiments, the error threshold ε of Algorithm 1 was set to zero (thus enforcing ν iterations). The parameters γ and ν were found by trail and error by looking for values that yielded a bang-bang control; this was motivated by the fact that the vast majority of optimal controls can be expected to belong to this class if the input sets are compact [39]. To avoid that the algorithm does not "overshoot" due to too large γ values, γ was initialized with a small number and increased iteratively, while comparing the control plots $u_1, \ldots, u_m$. The expert settings of CORA have been chosen as in the nonlinear tank model in the CORA manual [1]. A similar approach was taken in the case of Flow*.

Benchmarks description. The benchmark models are listed in the first column of Table 1. In addition to models taken from the literature, we considered a *binding model*, a biochemical model that can be parameterized in order to generate instances of increasing size.

		UTOPIC					*CORA*			*Flow**		
Model	*Size*	*Min*	*Max*	γ	ν	*Term.*	*Min*	*Max*	*Term.*	*Min*	*Max*	*Term.*
Brusselator	2	0.5780	1.1700	8	20	✓	-0.3931	2.3485	✓	-1.3139	3.1541	✓
Buckling	2	-1.1829	-0.9271	8	20	✓	-1.2039	-0.8817	✓	-1.4706	-0.6234	✓
JetEngine	2	-0.8717	-0.5813	100	50	✓	-1.0332	-0.3968	✓	-1.6446	-0.7168	✓
Van der Pol	2	0.9330	1.2300	10	100	✓	0.8690	1.3530	✓	0.7414	1.4785	✓
Robot Arm	4	0.597	0.738	1000	200	✓	0.5453	0.7767	✓	0.3719	0.9565	✓
Enzyme [17]	7	1.4773	1.7592	1000	50	✓	1.2473	1.9398	✓	-0.2956	3.4850	✓
Oscillator [50]	9	0.1242	0.1393	5	50	✓	0.1216	0.1422	✓	0.0910	0.1730	✓
Helicopter	28	2.3137	3.3600	2000	25	✓	2.1429	3.5551	✓	—		—[b]
Binding (2)	5	0.0454	0.0467	5	20	✓	0.0452	0.0482	✓	0.0428	0.0491	✓
Binding (3)	9	0.0242	0.0249	5	20	✓	0.0239	0.0265	✓	-0.0280	0.3149	✓
Binding (4)	17	0.0114	0.0117	10	20	✓	0.0112	0.0123	✓	-0.1816	0.3997	✓
Binding (5)	33	0.0104	0.0109	50	20	✓	—		TO	—		—[b]
Binding (6)	65	0.0100	0.0108	100	50	✓	—		TO	—		—[b]

Table 1: Numerical evaluation (with termination denoted by ✓). Unless specified, the benchmark models are taken from https://publish.illinois.edu/c2e2-tool/example/ (hybrid models were omitted, since Algorithm 1 applies to continuous systems only). TO: time-out (>10 h) during the symbolic computation of the Jacobian matrices; a) Divide-by-zero error; b) "Cannot compute flowpipes" error. The number of binding sites is shown within parenthesis alongside the model name.

The binding model is a simple protein-interaction network, taken from [13], where an enzyme, E, can form a complex with a substrate, P, according to a reversible reaction occurring at any of the n independent and identical binding sites of P. The model can be specified using the chemical reaction network

$$P_{(s_1,\ldots,s_i,0,s_{i+1},\ldots,s_n)} + E \xrightarrow{u_{a_i}} P_{(s_1,\ldots,s_i,1,s_{i+1},\ldots,s_n)}$$
$$P_{(s_1,\ldots,s_i,E,s_{i+1},\ldots,s_n)} \xrightarrow{u_{d_i}} P_{(s_1,\ldots,s_i,0,s_{i+1},\ldots,s_n)} + E,$$

where the first reaction models the binding of E to to the i-th binding site of P; instead, the second reaction refers to the unbinding. The parameters u_{a_i} and u_{d_i}, for $i = 1, \ldots, n$, give the association and dissociation constants for this model, respectively. The dynamics is visualized in Fig. 3, while the underlying ODEs are given, for any $n \geq 1$, by

$$\begin{aligned}\dot{x}_{P_s}(t) = &- \sum_{i:s(i)=1} u_{d_i}(t) x_{P_s}(t) - \sum_{i:s(i)=0} u_{a_i}(t) x_{P_s}(t) x_E(t) \\ &+ \sum_{i:s(i)=0} u_{d_i}(t) x_{A_{s+e^i}}(t) + \sum_{i:s(i)=1} u_{a_i}(t) x_{A_{s-e^i}}(t) x_E(t) \\ \dot{x}_E(t) = &\sum_s \Big[\sum_{i:s(i)=1} u_{d_i}(t) x_{P_s}(t) - \sum_{i:s(i)=0} u_{a_i}(t) x_{P_s}(t) x_E(t) \Big]\end{aligned}$$

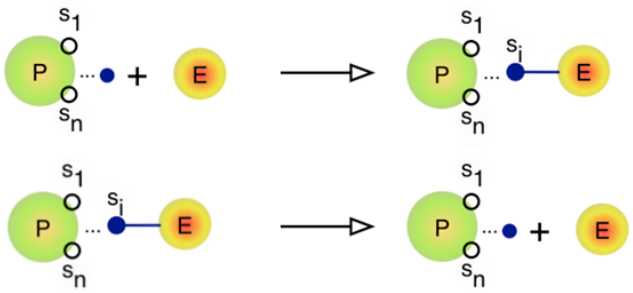

Fig. 3: Visualization of the binding unbinding pattern appearing in biochemical models [13]. The protein molecule P has n binding sites $s_1, \ldots, s_n$ that can be either empty or occupied by the molecule E.

In above ODEs, $s = (s_1, \ldots, s_n) \in \{0,1\}^n$ and $e^i \in \{0,1\}^n$ is such that $e^i_j = 1$ if $i = j$ and $e^i_j = 0$ otherwise. The ODE system of dimension $2^n + 1$ is obtained by interpreting the chemical reactions according to common law of mass action.

Results. The results reported in Table 1 confirm that the under-approximated reachable intervals are contained in the respective over-approximations. In particular, in the models where all techniques terminated (we fixed a time-out of 10 h on a machine equipped with a 2.7 GHz CPU core and 8 GB RAM), the under-approximation computed by UTOPIC is close to the tightest over-approximation. Since the binding models could neither be analyzed by CORA nor Flow* beyond the four-domain binding example (corresponding to 17 ODEs), larger model instances feature only results of UTOPIC.

We did not compare the runtimes because a fair comparison would be difficult for the following reasons:

(i) CORA and Flow* are over-approximation techniques, while UTOPIC is an under-approximation technique.
(ii) The runtimes may be sensibly affected by the choice of parameters.
(iii) CORA and Flow* approximate the whole reachable set across the whole time interval, while UTOPIC considers a one-dimensional linear projection of the state space at a given time point.

However, we report that the largest runtime of our algorithm was around ten minutes (of which eight were due to VNODE) for the binding model with 6 sites, indicating good scalability with increasing system sizes.

5 Conclusion

We presented a tool for the under-approximation of reachable sets of nonlinear ODEs with time-varying uncertainties. The approach was implemented in

UTOPIC, a fully automated tool-chain involving ERODE [12], MATLAB and VNODE [41]. An extensive evaluation demonstrated that UTOPIC provides tight under-approximations in benchmark models and scales to nonlinear systems with sizes as large as sixty-five. Overall, the results suggest that our algorithm can be a useful tool in the analysis of uncertain nonlinear systems.

Acknowledgement

Josu Doncel is supported by the Marie Sklodowska-Curie grant No. 777778, the Basque Government, Spain, Consolidated Research Group Grant IT649-13, and the Spanish Ministry of Economy and Competitiveness project MTM2016-76329-R. Max Tschaikowski is supported by a Lise Meitner Fellowship funded by the Austrian Science Fund (FWF) under grant No. M-2393-N32 (COCO).

References

1. M. Althoff. Cora 2016 manual.
2. M. Althoff. Reachability Analysis of Nonlinear Systems using Conservative Polynomialization and Non-Convex Sets. In *HSCC*, pages 173–182, 2013.
3. M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.
4. M. Althoff, C. Le Guernic, and B. H. Krogh. Reachable Set Computation for Uncertain Time-Varying Linear Systems. In *HSCC*, pages 93–102, 2011.
5. M. Althoff, O. Stursberg, and M. Buss. Reachability Analysis of Nonlinear Systems with Uncertain Parameters using Conservative Linearization. In *CDC*, pages 4042–4048, 2008.
6. E. Asarin, T. Dang, and A. Girard. Reachability Analysis of Nonlinear Systems Using Conservative Approximation. In *HSCC*, 2003.
7. Eugene Asarin and Thao Dang. Abstraction by projection and application to multi-affine systems. In *HSCC*, pages 32–47, 2004.
8. Luca Benvenuti, Davide Bresolin, Alberto Casagrande, Pieter Collins, Alberto Ferrari, Emanuele Mazzi, Alberto Sangiovanni-Vincentelli, and Tiziano Villa. Reachability computation for hybrid systems with ariadne. In *Proceedings of the 17th IFAC World Congress*, volume 41, pages 8960 – 8965, 2008.
9. Morten Bisgaard, David Gerhardt, Holger Hermanns, Jan Krcál, Gilles Nies, and Marvin Stenger. Battery-aware scheduling in low orbit: The gomx-3 case. In *FM*, pages 559–576, 2016.
10. L. Bortolussi and N. Gast. Mean Field Approximation of Uncertain Stochastic Models. In *DSN*, 2016.
11. L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Symbolic computation of differential equivalences. In *POPL*, 2016.
12. Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. ERODE: A tool for the evaluation and reduction of ordinary differential equations. In *TACAS*, pages 310–328, 2017.
13. Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Maximal aggregation of polynomial dynamical systems. *PNAS*, 114(38):10029–10034, 2017.

14. Taolue Chen, Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, and Aistis Simaitis. Automatic verification of competitive stochastic systems. In *TACAS*, pages 315–330, 2012.
15. Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An Analyzer for Non-linear Hybrid Systems. In *CAV*, pages 258–263, 2013.
16. Xin Chen, Sriram Sankaranarayanan, and Erika Ábrahám. Under-approximate flowpipes for non-linear continuous systems. In *FMCAD*, pages 59–66, 2014.
17. Gheorghe Craciun, Yangzhong Tang, and Martin Feinberg. Understanding bistability in complex enzyme-driven reaction networks. *PNAS*, 103(23):8697–8702, 2006.
18. T. Dang, C. L. Guernic, and O. Maler. Computing reachable states for nonlinear biological models. *TCS*, 412(21):2095 – 2107, 2011.
19. J. E. Dennis, Jr. and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, 1996.
20. Alexandre Donzé. Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems. In *CAV*, pages 167–170, 2010.
21. Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. Verification of Annotated Models from Executions. In *EMSOFT*, pages 26:1–26:10, 2013.
22. Parasara Sridhar Duggirala and Mahesh Viswanathan. Parsimonious, Simulation Based Verification of Linear Systems. In *CAV*, pages 477–494, 2016.
23. Chuchu Fan and Sayan Mitra. Bounded Verification with On-the-Fly Discrepancy Computation. In *ATVA*, pages 446–463, 2015.
24. Chuchu Fan, Bolun Qi, Sayan Mitra, Mahesh Viswanathan, and Parasara Sridhar Duggirala. Automatic reachability analysis for nonlinear hybrid models with C2E2. In *CAV*, pages 531–538, 2016.
25. Martin Fränzle, Christian Herde, Tino Teige, Stefan Ratschan, and Tobias Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *Journal on Satisfiability, Boolean Modeling and Computation*, 1:209–236, 2007.
26. A. Girard and G.J. Pappas. Approximate Bisimulations for Nonlinear Dynamical Systems. In *CDC*, pages 684–689, 2005.
27. Antoine Girard and Colas Le Guernic. Efficient reachability analysis for linear systems using support functions. In *Proceedings of the 17th IFAC World Congress*, volume 41, pages 8966–8971, 2008.
28. Antoine Girard, Colas Le Guernic, and Oded Maler. Efficient Computation of Reachable Sets of Linear Time-Invariant Systems with Inputs. In *HSCC*, pages 257–271, 2006.
29. Eric Goubault and Sylvie Putot. Forward inner-approximated reachability of non-linear continuous systems. In *HSCC*, pages 1–10, 2017.
30. Eric Goubault and Sylvie Putot. Inner and outer reachability for the verification of control systems. In *HSCC*, pages 11–22, 2019.
31. Donald Evan Kirk. *Optimal control theory: An introduction*. 1970.
32. Soonho Kong, Sicun Gao, Wei Chen, and Edmund M. Clarke. dReach: δ-Reachability Analysis for Hybrid Systems. In *TACAS*, pages 200–205, 2015.
33. A. B. Kurzhanski and P. Varaiya. Dynamic Optimization for Reachability Problems. *Journal of Optimization Theory and Applications*, 108(2):227–251, 2001.
34. A.B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis: internal approximation. *Systems & Control Letters*, 41(3):201 – 211, 2000.
35. Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. A New Class of Decidable Hybrid Systems. In *HSCC*, pages 137–151, 1999.

36. Kim Guldstrand Larsen. Validation, synthesis and optimization for cyber-physical systems. In *TACAS*, pages 3–20, 2017.
37. Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *COLT*, pages 1246–1257, 2016.
38. Meilun Li, Peter Nazier Mosaad, Martin Fränzle, Zhikun She, and Bai Xue. Safe over- and under-approximation of reachable sets for autonomous dynamical systems. In *FORMATS*, pages 252–270, 2018.
39. D. Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, 2011.
40. Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to Interval Analysis*. SIAM, 2009.
41. Nedialko S. Nedialkov. *Implementing a Rigorous ODE Solver Through Literate Programming*. 2011.
42. Truong Nghiem, Sriram Sankaranarayanan, Georgios E. Fainekos, Franjo Ivancic, Aarti Gupta, and George J. Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *HSCC*, pages 211–220, 2010.
43. M. S. Nikolskii. Convergence of the gradient projection method in optimal control problems. *Computational Mathematics and Modeling*, 18, 2007.
44. Stephen Prajna. Barrier certificates for nonlinear model validation. *Automatica*, 42(1):117–126, 2006.
45. Nacim Ramdani, Nacim Meslem, and Yves Candau. Computing reachable sets for uncertain nonlinear monotone systems. *Nonlinear Analysis: Hybrid Systems*, 4(2):263 – 278, 2010. IFAC World Congress 2008.
46. Joseph K. Scott and Paul I. Barton. Bounds on the reachable sets of nonlinear control systems. *Automatica*, 49(1):93–100, 2013.
47. Yasser Shoukry, Pierluigi Nuzzo, Indranil Saha, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia, George J. Pappas, and Paulo Tabuada. Scalable lazy SMT-based motion planning. In *CDC*, pages 6683–6688, 2016.
48. Ilya Tkachev and Alessandro Abate. A control lyapunov function approach for the computation of the infinite-horizon stochastic reach-avoid problem. In *CDC*, pages 3211–3216, 2013.
49. Max Tschaikowski and Mirco Tribastone. Approximate Reduction of Heterogenous Nonlinear Models With Differential Hulls. *IEEE Transactions Automatic Control*, 61(4):1099–1104, 2016.
50. José MG Vilar, Hao Yuan Kueh, Naama Barkai, and Stanislas Leibler. Mechanisms of noise-resistance in genetic oscillators. *PNAS*, 99(9):5988–5992, 2002.
51. Bai Xue, Martin Fränzle, and Naijun Zhan. Under-approximating reach sets for polynomial continuous systems. In *HSCC*, pages 51–60, 2018.
52. Bai Xue, Martin Frle, and Naijun Zhan. Inner-Approximating Reachable Sets for Polynomial Systems with Time-Varying Uncertainties. *CoRR*, 2018.
53. Bai Xue, Zhikun She, and Arvind Easwaran. Under-approximating backward reachable sets by polytopes. In *CAV*, pages 457–476, 2016.
54. Bai Xue, Qiuye Wang, Shenghua Feng, and Naijun Zhan. Over- and Under-Approximating Reachable Sets for Perturbed Delay Differential Equations. *CoRR*, 2019.
55. M. Zamani and R. Majumdar. A Lyapunov approach in incremental stability. In *CDC*, 2011.