Near Optimal Control With Reachability and Safety Guarantees.

Cees F. Verdier[a], Robert Babuška[a], Barys Shyrokau[a], and Manuel Mazo Jr.[a]

[a] The authors are with DCSC and Department of Cognitive Robotics, Delft University of Technology, Delft, The Netherlands,

Supported by NWO Domain TTW under the CADUSY project #13852.

Abstract

Control systems designed via learning methods, aiming at quasi-optimal solutions, typically lack stability and performance guarantees. We propose a method to construct a near-optimal control law by means of model-based reinforcement learning and subsequently verifying the reachability and safety of the closed-loop control system through an automatically synthesized Lyapunov barrier function. We demonstrate the method on the control of an anti-lock braking system. Here the optimal control synthesis is used to minimize the braking distance, whereas we use verification to show guaranteed convergence to standstill and formally bound the braking distance.

Keywords: Nonlinear optimal control, reinforcement learning, value iteration, verification, vehicle safety.

1 Introduction

Learning control systems typically rely on haphazard parameter tuning, without stability and performance guarantees of the learning algorithm and of the resulting controller. Recent years have witnessed an increased in interest in combining machine learning approaches with formal methods, including e.g. [1–3]. In this paper we propose a novel approach to automate formal controller synthesis through reinforcement learning, followed by a verification of reachability and safety guarantees of the synthesized controllers. We apply model-based reinforcement learning control design which returns a near-optimal controller described by an analytic expression [4]. This allows for verification using existing verification tools such as Flow* [5], CORA [6] and dReach [7], to name a few. In this work, we verify the closed-loop system by means of synthesis of a certificate function, similar to Lyapunov and barrier functions. We use a counter-example guided synthesis-like approach akin to [8–10], where candidate solutions are synthesized based on a finite set of sampled states, and are subsequently validated or disproved using an satisfiability modulo theories (SMT) solver: a tool capable of reasoning about first-order logic formulae based on a set of background theories. If a candidate is disproved, the SMT solver provides a counterexample which is used to enrich the set of sampled states. As in [10], we propose grammar-guided genetic programming to synthesize candidate solutions, which is an evolutionary computation method capable of evolving entire expressions. Here, no exact structure has to be pre-specified by the user, such as e.g. a fixed-order polynomial.

The proposed approach is demonstrated on the synthesis of an optimal controller for an Anti-lock Braking System (ABS), which actively controls the wheel dynamics during severe braking. Its purpose is to maximize braking performance while avoiding excessive wheel slip or wheel lock and so keeping the vehicle's ability to steer. We use an reinforcement learning-based control design to minimize the braking distance and formally verify convergence to standstill and bounds on the braking distance.

2 Preliminaries and problem definition

Given a system, we denote its state as $x \in \mathbb{R}^n$, a continuous time trajectory with $\xi : \mathbb{R} \to \mathbb{R}^n$ and a discrete state at time k with x_k. The interior of a set D is denoted by ∂D.

Consider a nonlinear system of the form

$$\dot{\xi}(t) = f(\xi(t), u(t)), \tag{1}$$

where $\xi(t) \in \mathcal{X} \subset \mathbb{R}^n$ and $u(t) \in \mathcal{U} \subset \mathbb{R}^m$ denote the state and input respectively. In this work we consider an optimal control design method for discrete-time systems, requiring system (1) to be discretized. To ensure aspects like reachability and safety are not lost for the original system, we formally verify the resulting control law w.r.t. original continuous time model.

2.1 Optimal control design

The discretized system (1) is described by the state transition function

$$x_{k+1} = f'(x_k, u_k) \tag{2}$$

with $x_k, x_{k+1} \in \mathcal{X}$ and $u_k \in \mathcal{U}$. This function is assumed to be available, but it does not have to be stated by explicit equations; it can be, for instance, a generative model given by a numerical simulation of complex differential equations. The control goal is specified through a *reward function* which assigns a scalar reward $r_{k+1} \in \mathbb{R}$ to each state transition from x_k to x_{k+1}:

$$r_{k+1} = \rho(x_k, u_k, x_{k+1})\,. \tag{3}$$

This function is defined by the user and typically calculates the reward based on the difference between the current state and a given constant reference state x_r that should be attained.

The goal is to find an (approximately) optimal control policy $\pi : \mathcal{X} \to \mathcal{U}$ such that in each state it selects a control action so that the cumulative discounted reward over time, called the return, is maximized:

$$R^\pi = E\Big\{\sum_{k=0}^{\infty} \gamma^k \rho\big(x_k, \pi(x_k), x_{k+1}\big)\Big\}\,. \tag{4}$$

Here $\gamma \in (0, 1)$ is a discount factor and the initial state x_0 is drawn uniformly from the state space domain $\mathcal{X}$ or its subset. Hence the considered control problem is:

Problem 2.1. *Design a control policy $\pi : \mathcal{X} \to \mathcal{U}$ such that the return is maximized.*

2.2 Formal verification

Given an optimal controller π, the next goal is to formally verify whether the continuous closed-loop system

$$\dot{\xi}(t) = f(\xi(t), \pi(\xi(t))) \tag{5}$$

satisfies a reachability and safety specification. Given a compact safe set $S \subseteq \mathcal{X}$, compact initial set $I \subset S$ and compact goal set G, the following specification is considered:

RWS *Reach while stay*: all trajectories starting in I eventually reach G, while staying within S:

$$\forall \xi(t_0) \in I, \exists T, \forall t \in [t_0, T] : \xi(t) \in S \wedge \xi(T) \in G. \tag{6}$$

We addresses the following problem:

Problem 2.2. *Given the compact sets* (S, I, G) *and closed-loop system* (5)*, verify that specification* RWS *is satisfied.*

This verification is done by means of automatic synthesis of a Lyapunov Barrier function as introduced in Section 3.2.

3 Methods

3.1 Optimal controller design

The return (4) is approximated by the value function $V^\pi : X \to \mathbb{R}$ defined as:

$$V^\pi(x) = E\Big\{\sum_{k=0}^{\infty} \gamma^k \rho\big(x_k, \pi(x_k), x_{k+1}\big)\Big| \, x_0 = x\Big\}. \tag{7}$$

An approximation of the optimal V-function, denoted by $\hat{V}^*(x)$, can be computed by solving the Bellman optimality equation

$$\hat{V}^*(x) = \max_{u \in \mathcal{U}} \Big[\rho\big(x, \pi(x), f'(x, u)\big) + \gamma \hat{V}^*\big(f(x, u)\big)\Big]. \tag{8}$$

To simplify the notation, in the sequel, we drop the hat and the star superscript: $V(x)$ will therefore denote the approximately optimal V-function.

To compute $V(x)$, we use the fuzzy V-iteration algorithm [11]. Given the process model (2) and the reward function (3), define the set $C = \{c_1, \dots, c_N\}$ of points on a regular grid in the state space. Further define a vector of triangular membership functions $\phi = [\phi_1(x), \dots, \phi_N(x)]^\top$ so that each $\phi_i(x)$ is centered at c_i, i.e., $\phi_i(c_i) = 1$ and $\phi_j(c_i) = 0$, $\forall j \neq i$. The membership functions are normalized so that $\sum_{j=1}^N \phi_j(x) = 1$, $\forall x \in \mathcal{X}$. Finally, define a finite set of discrete control input values $U = \{u^1, u^2, \dots, u^M\} \subset \mathcal{U}$.

The value function is approximated by the following basis-function expansion

$$V(x) = \theta^\top \phi(x)$$

where $\theta = [\theta_1, \dots, \theta_N]^\top \in \mathbb{R}^N$ is a parameter vector found through the following value iteration:

$$\theta_i \leftarrow \max_{u \in U} \left[\rho(c_i, u, f'(c_i, u)) + \gamma \theta^\top \phi\left(f'(c_i, u)\right)\right] \tag{9}$$

for $i = 1, 2, \dots, N$. This iteration is guaranteed to converge [11] and terminates when the following condition is satisfied:

$$||\theta - \theta^-||_\infty \leq \epsilon, \tag{10}$$

with θ^- the parameter vector calculated in the previous iteration and ϵ a user-defined convergence threshold. Fuzzy value iteration is very effective for second and third-order systems; computing the optimal value function is a matter of seconds. However, the computational and memory requirements grow exponentially and the method becomes impractical for systems above order four.

There are two principal ways to derive the control policy from the value function [4]. The first one is based on an online maximization of the Bellman optimality equation's right-hand side (hill-climbing policy), while the second one applies the Bellman equation off-line and uses basis functions to interpolate online (interpolated policy). Here we apply the latter method. For all states c_i, $i = 1, 2, \ldots, N$ compute off-line the optimal control action p_i:

$$p_i = \underset{u \in U}{\operatorname{argmax}} \; \left[\rho(c_i, u, f'(c_i, u)) + \gamma \theta^\top \phi\left(f'(c_i, u)\right)\right] \tag{11}$$

and collect the control actions in a vector: $p = [p_1, \ldots, p_N]^\top \in U^N$. In an arbitrary state x, the corresponding control action is then obtained by interpolation:

$$u = p^\top \phi\left(x\right) \tag{12}$$

where $\phi\left(x\right)$ are the same basis functions as defined for $V(x)$. An obvious advantage of this method is its computational simplicity: most computations are done off-line (vector p is actually obtained for free as a byproduct of the fuzzy value iteration algorithm) and the online interpolation is computationally cheap. Another advantage is that (12) directly produces continuous control actions. However, the control signal is not necessarily smooth and the interpolation can also result in a steady-state error. Therefore, in [4], we proposed a symbolic approximation method which is computationally effective and also yields smooth controls. A simplified version of this method is also applied here.

We build an analytic approximation of the policy in the following way. For a typical optimal control problem, the policy surface can be split into saturated parts where the control signal attains the minimal or maximal possible value, and a rather steep transition between the two parts. The transition is generally nonlinear, but often can be well enough approximated by a linear function. The overall policy is then described by:

$$u \quad = \quad \operatorname{sat}\big(Kx\big) \tag{13}$$

with K obtained by using linear regression on samples of the steep transition augmented with samples on the boundaries between the transition and the saturated hyper planes. The function $\operatorname{sat}(\cdot)$ defined as follows:

$$\operatorname{sat}(z) = \max\left(U_{\min}, \min\left(U_{\max}, z\right)\right)$$

For general systems for which such an approximation does not suffice, the aforementioned symbolic approximation in [4] can be used within the framework presented in this paper.

3.2 Verification through Lyapunov Barrier functions

The safety and reachability specification `RWS` can be verified indirectly by means of a Lyapunov barrier function (LBF), heavily inspired by control Lyapunov barrier functions, see e.g. [10] and the references therein.

Definition 3.1 (Lyapunov Barrier Function)**.** *A function $\mathcal{V} \in \mathcal{C}^1(S, \mathbb{R})$ is a Lyapunov Barrier Function w.r.t. the compact sets $S \subseteq \mathcal{X}$, $I, G \subseteq int(S)$ and system* (5) *if there exists a scalar $\gamma > 0$ such that*

$$\forall x \in I : \mathcal{V}(x) \leq 0, \tag{14a}$$

$$\forall x \in \partial S : \mathcal{V}(x) > 0, \tag{14b}$$

$$\forall x \in A \backslash G : \dot{\mathcal{V}}(x) \leq -\gamma, \tag{14c}$$

where $A := \{x \in S \mid \mathcal{V}(x) \leq 0\}$ and $\dot{\mathcal{V}}(x) = \langle \nabla \mathcal{V}(x), f(x, \pi(x)) \rangle$.

Note that the choice of γ is arbitrary, as shown in [10]. The existence of a LBF $\mathcal{V}$ implies that the closed-loop system satisfies specification `RWS`, as shown in the following theorem.

Theorem 1. *Given a system* (5)*, if there exists a LBF* $\mathcal{V}$ *w.r.t. compact sets* (S, I, G)*, then specification* `RWS` *holds.*

The proof is given in Appendix A. The possibility to prove `RWS` by the existence of a LBF motivates the automatic LBF synthesis. Our used method to automatically synthesize LBFs combines grammar-guided genetic programming with an SMT solver. Here the former is used to propose candidate LBFs, whereas the latter is used to formally verify candidate solutions or provide counterexamples.

3.3 Grammar-guided genetic programming

Similar to [10, 12], we use grammar-guided genetic programming, a variant of genetic programming [13]. Genetic programming is an evolutionary algorithm capable of synthesizing entire programs (in our case analytic expressions), rather than optimizing just over the parameters. This allows us to synthesize candidate LBFs, without restricting to a single parameterized structure.

Genetic programming utilizes a random population of candidate solutions that are encoded in way that allows for easy manipulation, typically an expression tree. This encoding is called the genotype, whereas the solution itself the phenotype. Given the random population, each candidate solution, also referred to as individual, is tested based on a fitness function. The returned fitness value measures the performance of the individual and is used to select fit individuals for recombination and adaptation by means of genetic operators. By applying the genetic operators, a new population is created with the hypothesis that these new individuals perform better than their ancestors. This procedure is repeated until a solution is found or a maximum number of generations is met.

As the name implies, the variant grammar-guided genetic programming utilizes a grammar to construct the genotype. This grammar can be used to restrict the search space and to bias the evolutionary search based on expert knowledge. We use a grammar in Backus-Naur form that is defined by the tuple $(\mathcal{N}, \mathcal{S}, \mathcal{P})$, where $\mathcal{N}$ denotes nonterminals, $\mathcal{S} \in \mathcal{N}$ the starting tree and $\mathcal{P}$ the production rules. An example of a grammar to construct polynomials is shown in Figure 1.

Based on a grammar, a genotype is grown by starting with the starting tree $\mathcal{S}$ and expanding all nonterminals in the leaves. This expansion is done by randomly selecting a subtree from the production rules corresponding to the nonterminal and placing it under the leave. Given the new tree, all leaves with nonterminals are expanded and this procedure is repeated until there are no more nonterminals in the leaves of the resulting tree. To prevent infinite length trees due to recursive production rules, a maximum recursion depth is defined, after which only the non-recursive rules can be selected. Given a genotype, its corresponding phenotype is obtained by replacing all nonterminals nodes by their underlying subtrees. An example of a fully grown genotype and its phenotype based on the grammar in Figure 1a are shown in Figure 1b and 1c.

In this work we use two genetic operators: crossover and mutation. In the former, two individuals are combined by interchanging two randomly selected subtrees with the same nonterminal and in the latter a random subtree under a nonterminal is replaced by a new subtree grown form the same nontermial. Finally, in each generation the constants present in the genotype are optimized using Covariance Matrix Adaptation Evolution Strategies (CMA-ES) [14].

The fitness function for LBF synthesis is constructed as follows. The three inequalities in (14) can be expressed as the following first order logic formula:

$$\varphi_i := \forall x \in X_i : g_i(x) \leq 0, \tag{15}$$

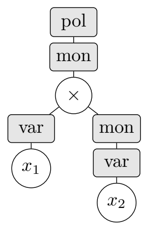
Nonterminals $\mathcal{N}$ and starting tree $\mathcal{S}$

$\mathcal{N} = \{\langle\text{pol}\rangle, \langle\text{mon}\rangle, \langle\text{var}\rangle, \langle\text{const}\rangle\}$,
$\mathcal{S} = \langle\text{pol}\rangle$

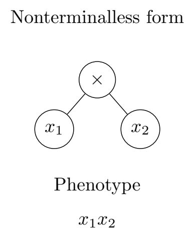
Production rules $\mathcal{P}$

$\langle\text{pol}\rangle ::= \langle\text{mon}\rangle \mid \langle\text{pol}\rangle + \langle\text{pol}\rangle$
$\langle\text{mon}\rangle ::= \langle\text{var}\rangle \mid \langle\text{var}\rangle \times \langle\text{mon}\rangle$
$\langle\text{var}\rangle ::= x_1 \mid x_2 \mid x_3$
$\langle\text{const}\rangle ::= \text{RandomReal} \in [-10, 10]$

(a) Grammar



(b) Genotype



(c) Phenotype

Figure 1: An example of a grammar to construct polynomials, a fully expanded genotype and the corresponding phenotype in tree form and analityc form.

with $i \in 1, 2, 3$. Given the standard form, two fitness metrics are considered. The first one is based on a finite number of test samples and is used to provide a search direction to the evolutionary search. The second metric is based on the outcome of an SMT solver, providing a Boolean answer to whether the inequallity formally holds.

Given the logic formula in (15) and a single point $x \in X_i$, we define the following error:

$$e_{\varphi_i}(x) := \max(g_i(x), 0). \tag{16}$$

Now for a finite set of test samples $\{x_1, \ldots, x_n\} \subseteq X_i$, the sample-based fitness metric is defined as:

$$\mathcal{F}_{\text{samp},\varphi_i} := (1 + \|[e_{\varphi_i}(x_1), \ldots, e_{\varphi_i}(x_n)]\|)^{-1}. \tag{17}$$

In this work we use the SMT solver dReal [15], as it is capable of reasoning about nonlinear inequalities over the reals. The fitness metric based on the SMT solver is defined as:

$$\mathcal{F}_{\text{SMT},\varphi_i} = \begin{cases} 1, & \text{if } \varphi_i \text{ is satisfied} \\ 0, & \text{otherwise} \end{cases}. \tag{18}$$

In case an inequality is not satisfied, the solver returns a small set in which it is violated. Samples from this violation set are added to the set of test samples X_i used in $\mathcal{F}_{\text{samp},\varphi_i}$, therefore refining the sample-based fitness evaluation.

The overall fitness function is defined as a weighted sum over the the sample-based and SMT-based fitness for all three conditions in (14):

$$\mathcal{F} := \frac{1}{6} \sum_{i=1}^{3} w_i \left(\mathcal{F}_{\text{samp},\varphi_i} + \mathcal{F}_{\text{SMT},\varphi_i}\right). \tag{19}$$

Here, the weights are sequentially defined as:

$$w_i = \lfloor w_{i-1} \mathcal{F}_{\text{samp},\varphi_{i-1}} \rfloor, \quad i \in \{2, 3\},$$

and $w_i = 1$. This weighting is motivated by the insight that prior to checking the condition on the derivative in (14c), the function should have the correct shape w.r.t. the initial set and the boundary set, as imposed by (14a) and (14b). Note that the full fitness function is equal to 1 if all conditions are formally verified.

3.4 LBF synthesis algorithm outline

The LBF synthesis algorithm undergoes the following steps, given a user-provided system (5), compact sets (S, I, G) and a grammar:

1. A population is created based on the grammar.
2. The parameters of each individual are optimized w.r.t. the sample-based fitness using CMA-ES.
3. The full fitness is computed.
4. Counterexamples returned from the SMT solver are added to the sets of sampled states.
5. The best individual is copied to the next generation.
6. New individuals are created by applying genetic operators on individuals which are selected using tournament selection [13], until a new full population is created.
7. Step 2 to 6 are repeated until an individual has the maximum fitness or a maximum number of generations is met.

4 Case-study: Anti-lock braking system

The control synthesis for an ABS system poses challenges due to the highly nonlinear and uncertain dynamic behavior of the wheel slip phenomenon. A longitudinal model of a corner vehicle is given by:

$$\begin{cases} \dot{v}(t) &= -\frac{1}{m}F(\kappa), \\ \dot{\omega}(t) &= \frac{r_t}{J}F(\kappa) - \frac{\sigma(\omega(t))}{J}u(t), \\ \dot{s}(t) &= v(t), \end{cases} \tag{20}$$

where $v(t)$ denotes the vehicle velocity, $\omega(t)$ the wheel angular velocity, $s(t)$ the braking distance, r_t the tire effective rolling radius, $u(t)$ the braking torque, m the corner vehicle mass and J the wheel moment of inertia. Moreover, $F(\kappa)$ is the longitudinal force due to the wheel slip κ:

$$F(\kappa) = mgd\sin(c\tan^{-1}\big(b(1-e)\kappa + e\tan^{-1}(b\kappa)\big), \tag{21}$$

with b, c, d and e road surface-specific constants and

$$\kappa = 1 - \frac{\omega(t)r_t}{v(t)}$$

the tire slip. Finally, $\sigma : \mathbb{R} \to \mathbb{R}$ is a continuous approximation of the signum function defined as

$$\sigma(x) = \tanh(100x). \tag{22}$$

In this case-study we use the parameters $J = 1.2$ kgm^2, $r_t = 0.305$ m, $m = 407.75$ kg and $g = 9.81$ m/s^2. We consider the slip force parameters $(b, c, d, e) = (55.56,\ 1.35,\ 0.4,\ 0.52)$, which correspond to wet asphalt for a water level of 3 mm [16]. Figure 2 shows the resulting force for different wheel slip values.

The choice of safe set, goal set and initial set are motivated as follows. According to the EU regulation N13 [17], for wet asphalt the maximum (initial) longitudinal velocity is 90 km/h (=25

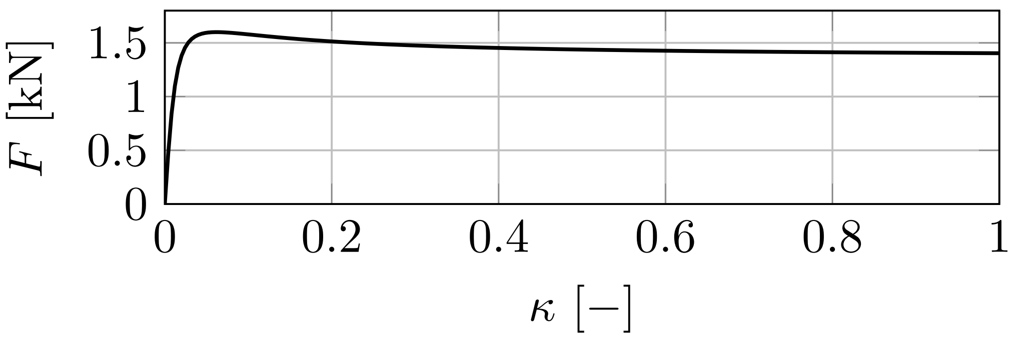


Figure 2: Longitudinal force vs. tire slip for a wet asphalt with a water level of 3 mm

m/s). The ABS is initialized of a slip angle of approximately 0 (i.e. $x_2 = x_1/r_t$) and is active until the longitudinal velocity meets the threshold of 5 km/h (= 5/3.6 m/s). Since the radius r_t can deviate slightly, we inflate the initial angular velocity to be bounded by $x_1/(r_t + \delta_r) \leq x_2 \leq x_1/(r_t - \delta_r)$. Finally, we impose an absolute maximum braking distance of 100 meters. This motivates the following choices for the safe set, initial set and goal set:

$$\begin{aligned}
S &= [0, 30] \times [-10, 30/r_t] \times [-10, 100]\,, \\
I &= \left\{ x \in S \;\middle|\; \frac{5}{3.6} \leq x_1 \leq 25,\ \frac{x_1}{r_t + \delta_r} \leq x_2 \leq \frac{x_1}{r_t - \delta_r}, \right. \\
&\qquad\qquad 0 \leq x_3 \leq 0.1 \Big\}\,, \\
G &= \{ x \in S \mid x_1 \leq 5/3.6 \}\,.
\end{aligned}$$

Given this safe set, the upper bound on the braking distance compared to the braking distance obtained from simulation is quite conservative. The bounds on the safe set could be chosen to be tighter, but this comes at the cost of longer computation times of the used SMT solver, assuming for the chosen bound a solution exists.

4.1 Controller design

For optimal control design, we use a discrete-time model obtained by numerically integrating the continuous-time dynamics (20) using the fourth-order Runge-Kutta method with the sampling period of $T_s = 0.001\,\mathrm{s}$. The state is the car velocity, $x_k = [v_k, \omega_k]^\top$, and the reward function is defined as:

$$r_{k+1} = \rho(x_k, u_k, x_{k+1}) = -x^\top Q x \tag{23}$$

with $Q = \mathrm{diag}(1, 0)$ a weighting matrix, specifying that the car velocity must reach zero, regardless of the wheel angular velocity.

The parameters of the fuzzy value iteration algorithm are listed in Table 1. The number of membership functions for each state variable was chosen quite large (31) in order to get a dense coverage of the state space domain of interest. The discount factor $\gamma = 0.999$ is selected close to one, so that not too much discounting takes place even at the end of a typical closed-loop transient which lasts about 1200 samples ($\gamma^{1200} \approx 0.3$).

The resulting policy is:

$$u(x) \quad = \quad \mathrm{sat}\big(k_1 x_1 + k_2 x_2 + k_0\big) \tag{24}$$

Table 1: Value iteration parameters

Parameter	Symbol	Value	Units
State domain	$\mathcal{X}$	$[0, 10] \times [0, 33]$	rad×rad/s
Num. of membership func.	N	$961 = 31 \times 31$	–
Discount factor	γ	0.9999	–
Convergence threshold	ϵ	0.001	–
Sampling period	T_s	0.001	s

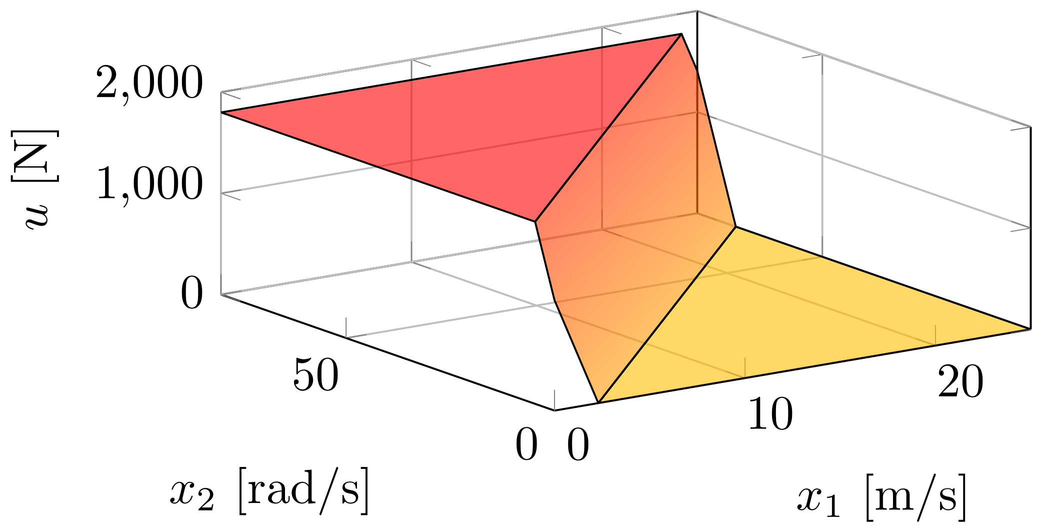


Figure 3: Piecewise linear policy (24) for wet asphalt with water level of 3 mm.

with $k_1 = 474.4$, $k_2 = 152.2$ and $k_0 = 1091.4$. This policy is shown in Figure 3.

For an initial condition of 90 km/h with a zero wheel slip and a sampling time of 0.001 seconds, we obtain from simulation a braking distance of 81.7874 meter. In comparison, in the case of a wheel lock, the braking distance is 90 meter.

4.2 Verification

The LBF synthesis is implemented in Mathematica 11.1 and performed on a desktop with Intel Xeon CPU E5-1660 v3 3.00 GHz using 14 parallel CPU cores. Given the specification sets S, I, G, we bias our search by having a grammar that imposes a template for the LBFs that consists of a polynomial plus a predefined barrier function. With the safe set written as $S = \Pi_{i=1}^{3}[\underline{s}_i, \overline{s}_i]$, we use a predefined barrier function $B : \mathbb{R}^3 \to \mathbb{R}$ of the form:

$$B(x) = \sum_{i=1}^{3} \frac{\langle \text{const} \rangle}{x_1 - \underline{s}_i + \varepsilon}, \tag{25}$$

where ε is a parameter that is chosen to be $\varepsilon = 0.001$. In our grammar, the starting symbol of the LBF is then selected to be

$$\mathcal{V}(x) = \mathcal{S} = \langle \text{const} \rangle + \langle \text{pol} \rangle + B(x), \tag{26}$$

Table 2: Statistics on the number of generations and total time for 5 successful LBF synthesis runs.

	Min	Max	Mean	SD
# generations	142	437	257.4	118.2
Total time [min]	51.3	315.6	161.1	104.4

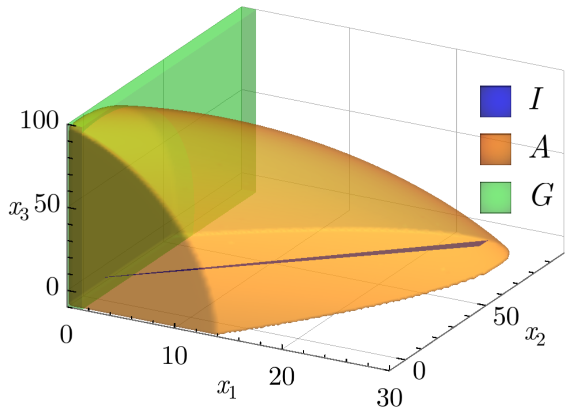


Figure 4: Sublevel set A of the synthesized LBF, initial set I and goal set G. Trajectories starting in I remain in A until they eventually reach G.

where ⟨const⟩ and ⟨pol⟩ denote nonterminals of a constant and polynomial. Besides the starting symbol, the remainder of the grammar is chosen to be as given in Figure 1a.

We consider a population of 28 individuals with a maximum of 500 generations and fix the number of CMA-ES generations to be 40. For the sample-based fitness, we start per inequality with a set of 100 samples, which can be complemented with up to 300 counterexamples, where a first-in-first out principle is used. The rates of the genetic operators are 0.5 for both crossover and mutation and the maximum tree recursion depth is chosen to be 6.

Synthesis is performed over 8 independent runs, in which 5 times a LBF was found before the maximum number of generations was met. For the 5 successful runs, the statistics on the number of generations and elapsed time is shown in Table 2. An example of a found solution is:

$$\begin{aligned}\mathcal{V}(x) =& -590553. + 985.64x_1 + 2303.02x_1^2 + 1230.37x_2\\ &- 1035.76x_1x_2 + 167.855x_2^2 - 1003.36x_3\\ &+ 94.5467x_1x_3 + 6.13646x_1^2x_3 + 69.6233x_3^2.\\ B(x) =& \frac{685.651}{0.001 + x_1} + \frac{621.366}{10.001 + x_2} + \frac{631.618}{10.001 + x_3}.\end{aligned}$$

The corresponding sublevel set A and sets I and G are shown in Figure 4. Note that set A can be seen as a forward invariant sublevel up until G is reached.

By the existence of an LBF, specification `RWS` holds w.r.t. the safe set, initial set and goal set. This implies that for all trajectories starting in the initial set, eventually the target velocity op 5km/h is reached and the braking distance up to that point is guaranteed to be below 100 meters. Note that if we select the initial set to be equal to the found sublevel set A, the conditions in (14) still hold, hence for all trajectories starting in A specification `RWS` holds.

5 Conclusions

We have demonstrated an automated approach to the design and verification for control systems. The proposed methodology combines reinforcement learning for optimal control design and automatic synthesis of LBF for formal verification of control specifications. The methodology has been demonstrated on the design of an ABS system.

In future work we intend to extend the verification to include uncertainties w.r.t. e.g. the mass and slip force parameters. These uncertainties can be introduced naturally into the LBF, similar to [9]. Moreover, typically controllers are implemented in a digital fashion, resulting in a sampled-data systems. We intent to extend the verification to include the sampled-data nature of a digital controller. Finally, the computed upper-bound of the braking distance is the same for all initial conditions. We plan to exploit the level-sets of the LBF to derive guaranteed bounds on the braking distance as a function of the initial condition.

A Proof of Theorem 1

First of all, by the continuity of $\mathcal{V}$ and the compactness of S, it follows that $\mathcal{V}$ is lower bounded on $S\backslash G$.

It follows from (14a) and the definition of A that for $\xi(t_0) \in I \subseteq A$. Using (14c) and the comparison theorem (see e.g. [18]), it follows that $\forall t > 0$, $\forall \xi(t) \in A\backslash G$: $\mathcal{V}(\xi(t)) \leq \mathcal{V}(\xi(t_0)) - \gamma(t - t_0) \leq -\gamma(t - t_0)$. Therefore, $\xi(t) \in A\backslash G$ implies that $\mathcal{V}(\xi(t))$ is decreasing and thus cannot reach the superlevel set $\mathcal{V}(x) > 0$. Therefore, by (14b), these trajectories cannot reach ∂S. Finally, since $\mathcal{V}(x)$ is lower bounded on the domain $A\backslash G \subseteq S\backslash G$, trajectories will decrease until in finite time $\xi(t)$ leaves $A\backslash G$ and can only enter G, therefore (6) holds. $\square$

References

[1] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in Neural Information Processing Systems 30.* Curran Associates, Inc., 2017, pp. 908–918.

[2] S. Pathak, L. Pulina, and A. Tacchella, "Verification and repair of control policies for safe reinforcement learning," *Applied Intelligence*, vol. 48, no. 4, pp. 886–908, Apr 2018.

[3] N. Fulton and A. Platzer, "Safe reinforcement learning via formal methods: Toward safe control through proof and learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[4] J. Kubalík, E. Alibekov, and R. Babuška, "Optimal control via reinforcement learning with symbolic policy approximation," in *Preprints 20th IFAC World Congress (IFAC-17)*, Toulouse, France, Jul. 2017.

[5] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *Computer Aided Verification.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 258–263.

[6] M. Althoff, "An introduction to CORA 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[7] S. Kong, S. Gao, W. Chen, and E. Clarke, “dreach: δ-reachability analysis for hybrid systems,” in *Tools and Algorithms for the Construction and Analysis of Systems.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 200–205.

[8] J. Kapinski, J. V. Deshmukh, S. Sankaranarayanan, and N. Arechiga, “Simulation-guided lyapunov analysis for hybrid dynamical systems,” in *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control.* New York, NY, USA: ACM, 2014, pp. 133–142.

[9] H. Ravanbakhsh and S. Sankaranarayanan, “Robust controller synthesis of switched systems using counterexample guided framework,” in *Embedded Software (EMSOFT), 2016 International Conference on.* IEEE, 2016, pp. 1–10.

[10] C. F. Verdier and M. Mazo Jr, “Formal synthesis of analytic controllers for sampled-data systems via genetic programming,” *arXiv preprint arXiv:1812.02711*, 2018.

[11] L. Buşoniu, D. Ernst, R. Babuška, and B. De Schutter, “Approximate dynamic programming with a fuzzy parameterization,” *Automatica*, vol. 46, no. 5, pp. 804–814, 2010.

[12] C. Verdier and J. M. Mazo, “Formal controller synthesis via genetic programming,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 7205 – 7210, 2017, 20th IFAC World Congress.

[13] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA, USA: MIT Press, 1992.

[14] N. Hansen and A. Ostermeier, “Completely derandomized self-adaptation in evolution strategies,” *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[15] S. Gao, S. Kong, and E. M. Clarke, “dReal: An SMT solver for nonlinear theories over the reals,” in *International Conference on Automated Deduction.* Springer, 2013, pp. 208–214.

[16] R. Gnadler, H.-J. Unrau, H. Fischlein, and M. Frey, *Ermittlung von My-Schlupf-Kurven an Pkw-Reifen*, ser. FAT-Schriftenreihe. FAT, Frankfurt/M., 1995, vol. 119.

[17] U. R. No13, “Uniform provisions concerning the approval of vehicles of categories M, N and O with regard to braking,” 2016.

[18] H. Khalil, *Nonlinear Systems*, ser. Pearson Education. Prentice Hall, 2002.