Modular Risk Assessment

A Compositional Verification Method for Approximated Automaton

Sugimura, Ryo

Award date:
2024

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering

Modular Risk Assessment: A Compositional Verification Method for Approximated Automaton

by

Sugimura R.

MSc Thesis

Thesis committee		**Graduation**	
Chair:	Dr. M. Lazar	Program:	Artificial Intelligence and Engineering Systems
Member 1:	Dr. S. Haesaert	Research group:	Control Systems
Member 2:	Dr. M. Chong	Thesis supervisor:	Dr. S. Haesaert
Member 3:	Dr. Z. Zhang	Date of defense:	04-07-2024
		Student ID:	1881760
		Study load (ECTS):	45
		Track:	Mobility

Modular Risk Assessment: A Compositional Verification Method for Approximated Automaton

Ryo Sugimura
Electrical Engineering
Eindhoven University of Technology
Eindhoven, Netherlands
r.sugimura@student.tue.nl

***Abstract*—Model checking is a formal verification method to assess whether a system satisfies specifications. It can be viewed as a form of risk assessment where specifications represent safety requirements. Model checking identifies violations in these safety specifications. A controller that minimizes risks can be synthesized based on model checking results. However, the formal verification method suffers from a state space explosion problem, where state space grows exponentially with the number of state variables. This work proposes a modular solution that 1) approximates an automaton of system behavior, which is 2) decomposed to create multiple subproblems, of which 3) the state variables are decoupled. The decomposition and decoupling address the state space explosion in model checking. The solution is applied to an overtaking scenario, demonstrating a reduction in computation time by a factor of 534 and memory usage by a factor of 155 compared to the formal verification method.**

***Index Terms*—risk assessment, model checking, controller synthesis, state space explosion**

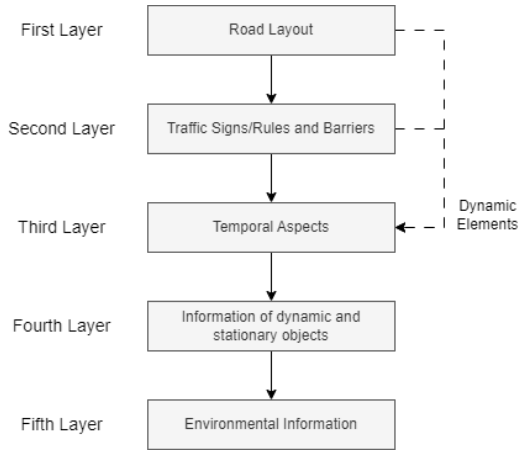

Fig. 1: Five layers of traffic scenario framework [5].

I. Introduction

The application and development of autonomous systems have seen a significant rise in diverse fields in recent years [1], [2]. In many of these fields, autonomous systems are required to perform complex tasks in environments where humans and autonomous systems coexist. Risk assessment is crucial to ensure safety in mixed environments. Risk assessment is a process to identify, analyze, and evaluate the potential hazards in an environment. One tool for risk assessment is model checking [3]. Model checking is a formal verification method to check whether the systems satisfy specifications. Specifications can include safety constraints addressing potential hazards. Assessing potential hazards in a system through model checking allows the synthesis of a controller that avoids those hazards. This method has been previously used to validate the safety of probabilistic intelligent vehicles [4].

This work focuses on applying model checking for risk assessment in the context of autonomous vehicles. These systems must be able to navigate a wide array of complex traffic scenarios. A traffic scenario is a detailed numerical representation of an environment where the autonomous vehicle operates, including its dynamics and objectives [5], [6]. According to the scheme introduced in Ma, Che, Li, *et al.* [5] and Bagschik, Menzel, and Maurer [6], the traffic scenario can be designed in five layers, as shown in Fig. 1. It is important to note that this five-layer structure is not a fact but rather one possible framework introduced [5], [6]. This work considers the first four layers in this five-layered description of traffic scenarios. With the representation of traffic scenarios as a formal model, model checking and controller synthesis can achieve risk assessment and control of autonomous vehicles.

While model checking is an effective tool for risk assessment, it frequently faces state space explosions. This problem refers to the exponential increase in the number of states as the number of state variables increases. Complex systems with high state dimensionality are more prone to state space explosions. Several solutions have been proposed to reduce the state space explosion problem, such as memory handling, heuristics-and-probabilistic reasoning, down-scaling the state space, bottom-up approach, and divide-and-conqueror approach [7]. Heuristic and probabilistic reasoning efficiently address the problem, especially when finding the exact solution is infeasible. The heuristic and probabilistic reasoning methods work by finding an approximate solution [8] or probabilistic representation of the large state space [9]. While it increases computational efficiency and applies to various systems, the solution may not be optimal due to its probabilistic nature and may be sensitive to the selected parameters. This work uses simplified and approximated models to reduce computational complexity while preserving essential system dynamics. Divide-and-conqueror approaches tackle the problem by

breaking down the global problem into subproblems. Each subproblem is solved independently and combined to analyze the global problem. The dependencies between the subproblems are preserved by defining relations between subproblems [10], [11]. While these relations make each subproblem manageable, relations may be complex. These approaches can simplify complex traffic scenarios by analyzing the movements as separate subproblems and then combining the solutions.

Heuristic and probabilistic reasoning approximate the model, reducing complexity and making otherwise infeasible problems solvable. In this study, the technique approximates an automaton, leading to feasible solutions and controllers. This paper also employs the divide-and-conqueror method to decompose the approximated automaton into subproblems, further reducing computational complexity.

This work presents a novel solution to model checking that combines heuristic and probabilistic reasoning and a divide-and-conqueror approach to leverage their strength. The essential contribution is a unified framework that addresses the state space explosion problem. First, the heuristic and probabilistic reasoning is utilized to find an approximate automaton that encapsulates the properties of the original. Second, the divide-and-conqueror process is used to decompose the automaton into subproblems. The decomposition allows an individual analysis of the subproblems. Furthermore, a machine learning model is applied for efficient risk assessment and controller synthesis. Not only does this reduce computational time and memory usage, but the efficiency of model checking and controller synthesis also allows this framework to be applied to complex systems.

This thesis is organized as follows. Sec. II introduces the problem formulation of model checking and controller synthesis. Sec. III explains the modular solution to address the state space explosion problem. Sec. IV investigates the effects of parameters and dynamics for model checking. Sec. V presents the numerical experiments to evaluate the modular solution. Finally, Sec. VI and VII discuss the results and conclude the thesis.

II. Problem Formulation

This section introduces the framework for model checking and controller synthesis, using an overtake scenario, a typical autonomous driving example, to illustrate the importance of risk assessment in planning an autonomous system. An overtake scenario is presented as a case study, using temporal logic to specify the vehicle's behavior. The traffic scenario is modeled as a Markov decision process, and probabilistic risk measures are introduced to quantify risk.

A. Overtake Scenario

Consider the following overtake scenario on the highway, illustrated in Fig. 2, where an ego vehicle intends to overtake the opponent vehicle. The opponent vehicle is considered to be moving at a constant speed. The first four layers are considered in this case study. The components in each of these layers are:

- First Layer: Two lanes, lane width (3.6 m), straight road,

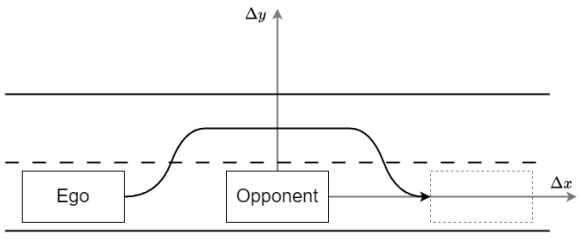

Fig. 2: Image of the overtake scenario.

- Second Layer: Speed limit (100 km/h),
- Third Layer: Constant road structure and rules,
- Fourth Layer: Position, velocity.

B. Dynamical System

The physical model of the vehicles can be represented as a dynamical system defining the evolution of traffic. The general form of the discrete-time dynamical systems showing the transitions between states over time is

$$M : \begin{cases} x_{t+1} = f(x_t, u_t, w_t) \\ y_t = g(x_t), \quad \forall t \in \{0, 1, 2, \dots\} \end{cases}, \tag{1}$$

initialized with x_0 and with state $x \in \mathbb{X}$, input $u \in \mathbb{U}$, stochastic noise $w \in \mathbb{W}$, and output $y \in \mathbb{Y}$ [12]. There are two measurable functions $f : \mathbb{X} \times \mathbb{U} \times \mathbb{W} \to \mathbb{X}$ and $g : \mathbb{X} \to \mathbb{Y}$. Furthermore, the stochastic noise w_t is assumed to be the realization of a Gaussian distribution $w_t \sim \mathcal{N}(0, \mathbb{I})$. The subscript t indicates the time step of the simulation. The past state and action pairs of the dynamical system can be expressed as

$$H_t := (x_0, u_0, x_1, u_1, \dots, u_{t-1}, x_t) \in (\mathbb{X} \times \mathbb{U})^t \times \mathbb{X} =: \mathbb{H}_t. \tag{2}$$

In the case of linear time-invariant (LTI) systems, the dynamical system can be simplified to

$$M : \begin{cases} x_{t+1} = Ax_t + Bu_t + B_w w_t \\ y_t = Cx_t \end{cases}, \tag{3}$$

with A and B matrices of appropriate dimensions [13]. Furthermore, the continuous states are gridded to finite states/inputs for abstraction [14]. The gridding is called finite state abstraction, where the finite state space $\hat{\mathbb{X}}$ is the center of those grids. The discretization of the input space $\mathbb{U}$ is applied to the system's abstraction. The size of the state grid and the number of discretized inputs define the abstraction resolution.

In this work, we will assume that the dynamical system can be decoupled into multiple dynamical systems to reduce the number of states in each system. For example, if a dynamical system was in a 2D space and states x_1 and x_2 were independent, the state evolution equation of the decoupled systems can be written as

$$\begin{cases} x_{1,t+1} = A_x x_{1,t} + B_x u_{1,t} + B_{w,x} w_{1,t} \\ x_{2,t+1} = A_y x_{2,t} + B_y u_{2,t} + B_{w,y} w_{2,t}, \quad \forall t \in \{0, 1, 2, \dots\}, \end{cases} \tag{4}$$

where the matrices $A_x, B_x, A_y, B_y, B_{w,x}, B_{w,y}$ are subject to decoupling properties such that dynamical system in Eq. (3)

can be decoupled into systems with x_1 and x_2. Furthermore, the noise $w_{1,t}$ and $w_{2,t}$ is considered independent.

The dynamical model of the vehicles concerned with the relative displacement of two vehicles is shown as

$$\begin{cases} \Delta x_{t+1} = \Delta x_t + \Delta v_{x,t}\Delta t + w_{1,t} \\ \Delta y_{t+1} = \Delta y_t + \Delta v_{y,t}\Delta t + w_{2,t} \\ \Delta v_{x,t+1} = \Delta v_{x,t} + a_x\Delta t + w_{3,t} \\ \Delta v_{y,t+1} = \Delta v_{y,t} + a_y\Delta t + w_{4,t}, \end{cases} \tag{5}$$

where states $(\Delta x_t, \Delta y_t, \Delta v_{x,t}, \Delta v_{y,t})$ are the difference in position and velocity at time t centered around the opponent vehicle. The input is the acceleration of the ego vehicle a_x and a_y. The $w_{i,t}$ is the noise of the states with $w_{i,t} \sim \mathcal{N}(0, \sigma_i)$, and each of the noise is considered independent realizations. The physical limits for this scenario are

$$\mathbb{X}: \begin{cases} |\Delta x_t| \le 15 \text{ [m]} \\ -1.8 \le \Delta y_t \le 5.4 \text{ [m]} \\ |\Delta v_{x,t}| \le 11 \text{ [m/s]} \\ |\Delta v_{y,t}| \le 1 \text{ [m/s]}, \end{cases} \tag{6}$$

where the range of the relative positions is based on the size of the highway and inter-vehicle distance on a highway. For x and y velocities, the range is decided based on safe human driving. The input limits can be written as:

$$\mathbb{U}: \begin{cases} |a_x| \le 1.5 \ [m/s^2] \\ |a_y| \le 0.5 \ [m/s^2], \end{cases} \tag{7}$$

based on the safe behavior of a human driver. Furthermore, the dynamical system can be decoupled by separating the states in the x and y directions.

C. Specifications

Specifications can be written using linear temporal logic (LTL), a mathematical language to describe constraints in linear time. This work uses syntactically co-safe LTL (scLTL) [15], a subset of LTL, for formulating specifications with advantages in calculating satisfaction by considering finite steps. If atomic propositions are denoted as $AP = \{p_1, \dots p_n\}$, the alphabet can be denoted as $\Sigma = 2^{AP}$ with letter $l \in 2^{AP}$. The specification φ is defined as

$$\varphi ::= p | \neg p | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | \varphi_1 \mathsf{U} \varphi_2 | \bigcirc \varphi, \tag{8}$$

with operators negation $\neg$, conjunction $\wedge$, disjunction $\vee$, until U and next $\bigcirc$, and specifications φ_1 and φ_2 [16]. The semantics of the LTL for the word $\boldsymbol{\pi}_t$ are defined as follows. A word $\boldsymbol{\pi}_t$ is an infinite sequence of letters $l_i \in \Sigma$, as shown as $\boldsymbol{\pi}_t = l_t\, l_{t+1}\, l_{t+2} \dots$. An atomic proposition $\boldsymbol{\pi}_t \vDash p$ if $p \in l_t$. A negation $\boldsymbol{\pi}_t \vDash \neg p$ holds if $p \notin l_t$. A conjunction $\boldsymbol{\pi}_t \vDash \varphi_1 \wedge \varphi_2$ holds if both $\boldsymbol{\pi}_t \vDash \varphi_1$ and $\boldsymbol{\pi}_t \vDash \varphi_2$ are satisfied. A disjunction $\boldsymbol{\pi}_t \vDash \varphi_1 \vee \varphi_2$ holds if either $\boldsymbol{\pi}_t \vDash \varphi_1$ or $\boldsymbol{\pi}_t \vDash \varphi_2$ is satisfied. The until operator $\boldsymbol{\pi}_t \vDash \varphi_1 \mathsf{U} \varphi_2$ holds if there is $j \in \mathbb{N}$ such that $\pi_{t+j} \vDash \varphi_2$ and for all $i \in \mathbb{N}$, $0 \le i < j$ we have $\pi_{t+i} \vDash \varphi_1$. The next operator $\boldsymbol{\pi}_t \vDash \bigcirc \varphi$ holds if $\pi_{t+1} \vDash \varphi$ is satisfied. The time-bounded temporal operators can be written as are

$$\varphi ::= \Box^{\le I}(\varphi_1) | \Diamond^{\le I}(\varphi_1) | \varphi_1 \mathsf{U}^{\le I} \varphi_2, \tag{9}$$

where the $\Box$, $\Diamond$ and U represent always, eventually and until, respectively. The variable I represents the time bound for the temporal logic specification. The semantics for the temporal operators are

$$\boldsymbol{\pi}_t \vDash \Box^{\le I}\varphi_1 \text{ holds if } \forall i \le I,\ \boldsymbol{\pi}_{t+i} \vDash \varphi_1, \tag{10}$$

$$\boldsymbol{\pi}_t \vDash \Diamond^{\le I}\varphi_1 \text{ holds if } \exists i \le I,\ \boldsymbol{\pi}_{t+i} \vDash \varphi_1, \tag{11}$$

$$\boldsymbol{\pi}_t \vDash \varphi_1 \mathsf{U}^{\le I} \varphi_2 \text{ holds if } \begin{cases} \exists j \le I,\ \boldsymbol{\pi}_{t+j} \vDash \varphi_2, \\ \boldsymbol{\pi}_{t+i} \vDash \varphi_1,\ \forall i, 0 \le i < j. \end{cases} \tag{12}$$

Combined atomic propositions and operators can represent more complex specifications.

The atomic propositions can be geometrically visualized by defining a polytope within the state space. A polytope $P \subseteq \mathbb{R}^d$ is a geometric object that is described in two ways: a convex hull of finite points ($\mathcal{V}$-polytope) or an intersection of many closed halfspaces in $\mathbb{R}^d$ ($\mathcal{H}$-polytope) [17]. In this work, the atomic proposition is assumed to have $\mathcal{H}$-polytope representation, where the polytope P is defined as:

$$P = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{A}\mathbf{x} \le \mathbf{b}\} \text{ for some } \mathbf{A} \in \mathbb{R}^{m \times d}, \mathbf{b} \in \mathbb{R}^m, \tag{13}$$

where $\mathbf{x}$ is the states inside the halfspace. The polytope defined by the letter l is denoted as $P(l)$, and the set of states inside the polytope $P(l)$ is denoted as $S(l)$.

A scLTL specification φ can be translated to deterministic finite automaton (DFA), which can be used to verify if a controller satisfies specifications. A DFA is defined as $\mathscr{F} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$ constructed with a finite set of states $\mathcal{Q}$, an alphabet Σ, a transition function $\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$, a start state $q_0 \in \mathcal{Q}$, and a final state $F \in \mathcal{Q}$. A sink state $q_s \in (\mathcal{Q} \backslash F)$ has only incoming transitions. In DFA, a transition from state q_i to state q_j on input letter l denoted as $q_i \xrightarrow{l} q_j$ is formally defined by the transition function $\delta(q_i, l) = q_j$. A word w_t is accepted by the DFA if the sequence of states satisfies the specification φ. Additionally, if DFA is translated from scLTL, then the word is accepted by the DFA if it satisfies the scLTL specification. Furthermore, the DFA can be visualized as a directed graph where nodes represent the states $\mathcal{Q}$, directed edges represent the transitions δ labeled with a letter $l \in \Sigma$, empty incoming edge denoted the starting state q_0, double circle represent the final state F, and a node with no outgoing edges represents the sink state q_s.

The atomic propositions of the overtake scenario are written as:

$$\begin{aligned} &p_1 : |\Delta y_t| \le 1.8, && \text{Right lane.} \\ &p_2 : 5 \le \Delta x_t \le 20, && \text{In front of opponent.} \\ &p_3 : -20 \le \Delta x_t \le -5, && \text{Behind opponent.} \end{aligned} \tag{14}$$

The objective is written as $\Diamond(p_1 \wedge p_2)$ and keeping a safe distance is written as $\Box(\neg p_1 \vee (p_2 \vee p_3))$. In the overtake, the safety specification needs to hold for all time steps before

satisfying the objective. Therefore, the specification can be written as $(\neg p_1 \vee (p_2 \vee p_3))\mathsf{U}(p_1 \wedge p_2)$. The controller that satisfies this specification is considered a safe overtake. Table I shows the combination of atomic propositions defining each letter. Given the specification $(\neg p_1 \vee (p_2 \vee p_3))\mathsf{U}(p_1 \wedge p_2)$,

TABLE I: Atomic proposition combination for each letter (ex. $l_1 \coloneqq p_1 \wedge \neg p_2 \wedge p_3$).

l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
p_1	$\neg p_1$	$\neg p_1$	$\neg p_1$	$\neg p_1$	p_1	p_1	p_1
$\neg p_2$	$\neg p_2$	$\neg p_2$	p_2	p_2	$\neg p_2$	p_2	p_2
p_3	p_3	$\neg p_3$	$\neg p_3$	p_3	$\neg p_3$	$\neg p_3$	p_3

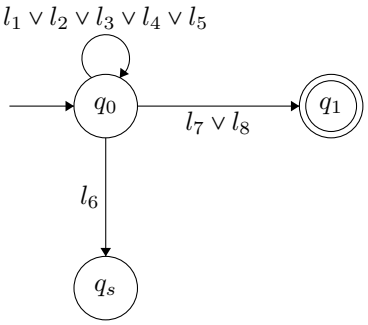

Fig. 3: Original DFA of overtake scenario.

the specification can be translated into a DFA shown in Fig. 3. The actions that trigger the transitions are the letters $l_1, \ldots, l_8$ defined in Table I. This DFA has states $\mathcal{Q} = \{q_0, q_1, q_s\}$ where q_0 is the starting state, q_1 is the final state, and q_s is the sink state. Furthermore, the letters define a polytope in the state space as shown in Fig. 4. The letters l_5 and l_8 are empty sets since they intersect two non-overlapping polytopes in the x direction.

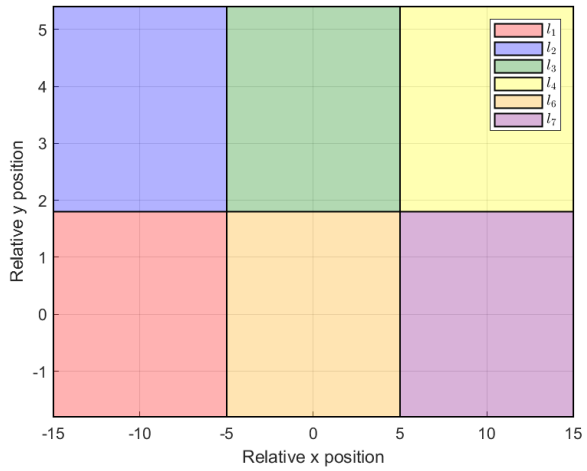

Fig. 4: Polytope representation of letters in alphabet.

D. Markov Decision Process

A Markov decision process (MDP) can model the traffic and how the vehicle makes decisions. An MDP is a mathematical framework for decision-making that incorporates probabilistic and non-deterministic choices [3], [18] that adheres to Definition 1.

Definition 1. *A Markov decision process is denoted as* $\mathscr{M} = (S, Act, \mathbf{P}, d_{init}, AP, L)$ *where*

- S *is a set of states,*
- Act *is a set of actions,*
- $\mathbf{P} : S \times Act \times S \to [0,1]$ *is a probabilistic transition function such that for a certain state* $s \in S$ *and action* $\alpha \in Act$, $\sum_{s' \in S} \mathbf{P}, (s, \alpha, s') \in \{0,1\}$ *holds,*
- $d_{init} : S \to [0,1]$ *is a initial state probability such that* $\sum_{s \in S} d_{init}(s) = 1$,
- AP *is the set of atomic propositions,*
- $L : S \to 2^{AP}$ *is a labeling function.*

This work assumes that the MDP models the dynamical system. Furthermore, this work assumes that Assumption 1 holds.

Assumption 1. *The MDP can be decoupled into smaller, independent MDPs* $\mathscr{M}_1, \mathscr{M}_2, \ldots, \mathscr{M}_n$ *by partitioning its state space* $\mathbb{X}$ *into disjoint subsets* $\mathbb{X}_1, \mathbb{X}_2, \ldots, \mathbb{X}_n$ *and its action space* $\mathbb{U}$ *into disjoint subsets* $\mathbb{U}_1, \mathbb{U}_2, \ldots, \mathbb{U}_n$.

This assumption allows the decoupled MDPs $\mathscr{M}_1, \mathscr{M}_2, \ldots$ to be solved independently. In the case of 2D state space, the MDP can be decoupled into MDPs for x and y denoted as $\mathscr{M}_x$ and $\mathscr{M}_y$.

The solution to the Markov decision process is a control strategy called policy. The policy acts as a decision-making model for agents that decide the next action based on the sequence of state action pairs. The general form of the policy can be written as $\mu = \mu_0, \mu_1, \mu_2, \mu_3, \ldots$, where $\mu_t : \mathbb{H}_t \to \mathbb{U}$. The overall policy is the set of policies at each time interval that maps the history to the actions. A policy can have two attributes: Markov property [19] and stationarity [20]. The Markov property can be written as $\mu_t : \mathbb{X} \to \mathbb{U}$, where the policy maps current states to actions but is not fixed over time. Whereas, for the stationary policy, the policy does not change over the time step as shown in $\mu = \mu_s, \mu_s, \mu_s, \mu_s, \ldots$, where $\mu_s : \mathbb{X} \to \mathbb{U}$. In this work, we will use a policy that follows Markov property and is considered stationary, such that it is fixed over time and only dependent on the past state. An optimal controller or policy for an MDP defines the behavior of the vehicles in a traffic environment modeled by the MDP.

E. Risk Quantification

A risk measure has to be chosen to assess the risk in traffic scenarios. The risk measures include a collision-based measure and a combination of traffic rules and driver intent preference. The mapping of the risk measure is shown as $R : \mathbb{X}^t \to \mathbb{R}$, where it maps the sequence of past states to risk values. The optimal control strategy would be to choose the policy that minimizes the risk, as shown in

$$\min_{\mu} R. \tag{15}$$

This paper quantifies risk with satisfaction probability of specification φ as

$$R = \mathbb{P}(w \vDash \varphi), \tag{16}$$

where w is the word. The satisfaction probability can be utilized as a risk measure by incorporating safety specifications. A high level of satisfaction with a safety specification indicates that the system is safe. Choosing a control strategy based on this risk measure is to maximize the probability that a policy satisfies a certain specification, as shown in

$$\max \mathbb{P}(w \vDash \varphi), \tag{17}$$

where w is the word defined by the policy and φ is the specification. One part of risk assessment is to find a controller for risk mitigation. Risk mitigation is selecting a policy or control strategy aimed at minimizing risk. Finding the policy or control strategy that satisfies Eq. (17) will mitigate the risks in a scenario.

F. Problem Statement

The evolution of the environment is modeled as a dynamical system M. The agent operating within this environment follows a behavior defined by specification φ. A Markov decision process $\mathscr{M}$ is used to model agents' decision-making process in the dynamical system. The risk is quantified with satisfaction probability $\mathbb{P}(w \vDash \varphi)$, where w is the word under a given policy μ. Model checking and controller synthesis aim to calculate this risk and find a policy μ for MDP $\mathscr{M}$ that maximizes satisfaction as expressed by

$$\mu = \max \mathbb{P}(w \vDash \varphi). \tag{18}$$

However, formal verification methods suffer from the state space explosion problem, where the state space expands exponentially as the state dimensionality increases. The problem comes from finite state abstraction, where the state space is gridded, and its size is calculated as

$$\prod_{i=1}^{N} g_i, \tag{19}$$

where g_i is the number of grids in i-th state variable out of N total state variables. This problem leads to longer computational time and large memory usage, thus making it inapplicable to complex systems with more state variables. How can the efficiency of model checking and controller synthesis be improved?

The proposed solution is a modularized solution that works in three steps: approximation of DFA, decomposition of DFA, and parameter selection using machine learning. First, the approximated DFA $\hat{\mathscr{F}}$ is constructed from the original one $\mathscr{F}$. Second, the approximated DFA $\hat{\mathscr{F}}$ is decomposed into subgraphs or subproblems $\hat{\mathscr{F}}_1, \hat{\mathscr{F}}_2, \dots$ and the dynamics are decoupled to make a set of multiple MDPs $(\mathscr{M}_{1,x}, \mathscr{M}_{1,y}, \mathscr{M}_{2,x}, \mathscr{M}_{2,y}, \dots)$. Finally, a machine learning model is used to select the parameter for optimal controller synthesis within decomposed MDPs. Based on the time bounds, the specification and atomic proposition are updated. The model checking is done for each decoupled MDP.

III. Modular Solution

The state space explosion limits the applicability of model checking and controller synthesis to complex systems. This section presents a modular solution to efficiently model check and synthesize controllers for complex systems by reducing this problem.

A. Solution Overview

One solution is intuitively splitting the global problem into parts with its specifications into subproblems, a smaller problem within the original problem [21], [22]. However, the intuitive approach depends on understanding the scenario and its complexity. A more systematic approach is the modular solution, a compositional verification method on an approximated model decomposing the deterministic finite automaton (DFA) into subproblems.

The modular solution is applied following these steps as shown in Fig. 5. First, an enlarged DFA is constructed by splitting the states of the original DFA. Then, a path in the enlarged DFA is selected to obtain the subgraph for model checking. Next, find the reach-avoid patterns in the DFA. Then, formulate the decoupled and time-bounded scLTL specifications. Next, value iteration obtains value function with different time-bound to model check each pattern. Finally, the machine learning model finds the optimal time-bound for controller synthesis and implementation.

B. Assumptions

A few assumptions exist for the problem set to apply the modular solution. The assumptions are decouplable dynamical systems and independent atomic propositions. The assumption for the decouplable dynamical system is shown in Assumption 2.

Assumption 2. *The dynamical system needs to be decouplable into smaller dimensions. Two conditions must be met simultaneously so the dynamical system can be decoupled.*

1) *The A matrix of the state space model is in block diagonal form, written as*

$$A = \begin{bmatrix} A_1 & & 0 \\ & \ddots & \\ 0 & & A_N \end{bmatrix}, \tag{20}$$

where A_i is the $n_i \times n_i$ square matrix.

2) *The B matrix of the state space model has to be in the form of*

$$B = \begin{bmatrix} B_1 & & 0 \\ & \ddots & \\ 0 & & B_n \end{bmatrix}, \tag{21}$$

where B_i is a $n_i \times m_i$ matrix based on the square matrix of block diagonal form.

Another option is a Jordan canonical form J for the A matrix. This condition is written as

$$\exists\, J : P^{-1} J P = A, \tag{22}$$

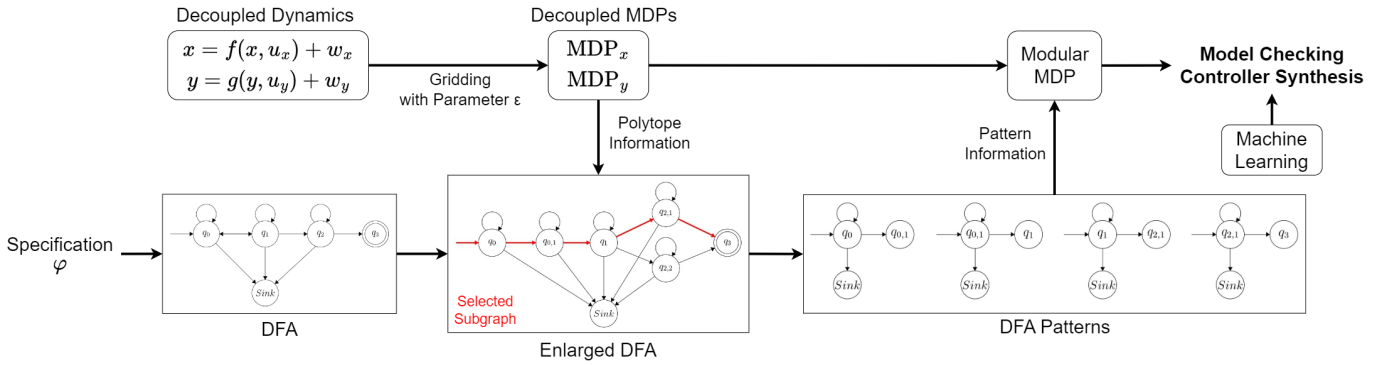

Fig. 5: Overview of the modular solution for model checking and controller synthesis.

where J is the Jordan matrix and P is the change-of-basis matrix.

This assumption allows the dynamical system and the Markov decision process to be decoupled, thus reducing the number of state variables. The independence of atomic propositions is shown in Assumption 3.

Assumption 3. *The atomic propositions are defined independently for each dimension, meaning they cannot depend on variables from other dimensions. If a polytope defined by the atomic proposition $P(p)$ is written with $\mathcal{H}$-polytope representation as*

$$P(p) = \{x \in \mathbb{R}^d \mid \mathbf{H}\mathbf{x} \leq \mathbf{b}\}, \tag{23}$$

where x is the states, the matrix $\mathbf{H}$ has to have the form of

$$\mathbf{H} = \begin{bmatrix} 0 & \cdots & \mathbf{H}_i & \cdots & 0 \end{bmatrix}, \tag{24}$$

where $\mathbf{H}_i$ is a square matrix.

This assumption is needed for the letter l to be written as a conjunction of atomic propositions p in each dimension as

$$l \coloneqq p_1 \wedge p_2 \wedge \cdots \wedge p_N, \tag{25}$$

where p_i is one of the atomic propositions in the N decoupled dynamical systems.

C. Construction of Enlarged DFA

The enlarged DFA, as defined in Definition 2, is obtained by splitting the states of the original DFA.

Definition 2. *Given the original DFA $\mathscr{F} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$, the enlarged DFA is defined as $\hat{\mathscr{F}} = (\hat{\mathcal{Q}}, \hat{\Sigma}, \hat{\delta}, \hat{q}_0, \hat{F})$ where*

$$\begin{aligned} \mathcal{Q} &\subseteq \hat{\mathcal{Q}}, \\ \Sigma &= \hat{\Sigma}, \\ q_0 &= \hat{q}_0, \\ F &= \hat{F}, \end{aligned} \tag{26}$$

with appropriate transition function $\hat{\delta}$ obtained by Algorithm 1.

This process is guided by facet $\mathcal{F}$, a face of dimension $d-1$ defined as

$$\mathcal{F} = P \cap \{\mathbf{x} \in \mathbb{R}^d : \mathbf{c}\mathbf{x} = c_0\}, \tag{27}$$

with $\mathbf{c}\mathbf{x} \leq c_0$ being a valid inequality for polytope P [17]. Based on the DFA structure, the state-splitting algorithm can be applied to obtain the enlarged DFA. The state-splitting algorithm is based on two cases:

1) State space is gridded by polytopes, with more than one connected polytope path w_p.
2) Polytopes are spread in state space with no connected polytope path w_p.

The connected polytope path w_p defines which case the problem is classified as and is given in Definition 3.

Definition 3. *A connected polytope path w_p is a word of finite length satisfying the three conditions shown as*

$$w_p = l_0\, l_1 \ldots l_N \text{ such that } \begin{cases} \delta(q_{i-1}, l_0) = q_i \\ \delta(q_i, l_N) = q_{i+1} \\ \exists\, \mathcal{F} : P(l_i) \cap P(l_{i+1}) = \mathcal{F} \end{cases}, \tag{28}$$

where q_i is a DFA state, $l_i \in \Sigma$ is a letter, $P(l_i)$ is the polytope defined by letter l_i, and $\mathcal{F}$ is a facet. In the case of the first DFA state, the letter l_0 in the first condition is considered the starting polytope.

If a connected polytope path exists, the third condition in Eq. (28) is also considered the transition condition. In the context of an overtake scenario captured by DFA (Fig. 3) and characterized by the polytope (Fig. 4), a feasible word subject to the definition of connected polytope path reads

$$w_p = l_1\, l_2\, l_3\, l_4\, l_7. \tag{29}$$

Based on the connected polytope path, these steps are applied to split the states using Algorithm 1. In the case of the overtake scenario, only one state q_0 needs to be considered. Since there are five letters in the self-loop, there will be four additional states $q_{0,1}, \ldots, q_{0,4}$. Looking at the first two letters in the connected polytope path w_p, we can see the polytopes have a facet $\mathcal{F}$ as an intersection as

$$\begin{aligned} &P(l_1) \cap P(l_2) = \mathcal{F}, \\ &\text{where } \mathcal{F} = \{s \in \mathbb{R}^2 \mid -20 \leq \Delta x \leq -5, \Delta y = 1.8\}. \end{aligned} \tag{30}$$

Algorithm 1 State-Splitting Algorithm

Remove $\{l \in \Sigma \mid P(l) = \varnothing\}$ from Σ and δ
for q_i in $\mathcal{Q} \backslash (q_0 \cup q_s)$ **do**
  if w_p exists **then**
    Number of additional node n = $|\{l \in \Sigma \mid \delta(q_i, l) = q_i\}| - 1$
    New states $\mathcal{S} = \{q_{i,1}, \ldots, q_{i,n}\}$
    Update states $\mathcal{Q} = \mathcal{Q} \cup \mathcal{S}$
    for letter pair $(l_j, l_k) \in \{l \in \Sigma \mid \delta(q_i, l) = q_i\}^2$ **do**
      if $P(l_j) \cap P(l_k) = \mathcal{F}$ **then**
        Update transition function δ
      end if
    end for
  else if w_p doesn't exist **then**
    Number of additional node m = $|\{l \in \Sigma \mid \delta(q_i, l) = q_{i+1}\}|$
    New states $\mathcal{S} = \{q_{i+1,1}, \ldots, q_{i+1,m}\}$
    Update states $\mathcal{Q} = \mathcal{Q} \cup \mathcal{S}$
    for $j = 1$ to m **do**
      Update transition $\delta(q_i, l_j) = q_{i+1,j}$
      If self loop exists, update $\delta(q_{i+1,j}, l_j) = q_{i+1,j}$
    end for
  end if
end for
Construct enlarged DFA

Following Algorithm 1, a new transition can be added as follows

$$\begin{aligned} \delta(q_0, l_2) &= q_{0,1} \\ \delta(q_{0,1}, l_1) &= q_0. \end{aligned} \tag{31}$$

These steps are iterated for the remaining letter combination in the connected polytope path to construct the enlarged DFA for the overtake scenario (Fig. 6). The additional states added through Algorithm 1 is $q_{0,1}$, $q_{0,2}$ and $q_{0,3}$ being an extension of the first state q_0.

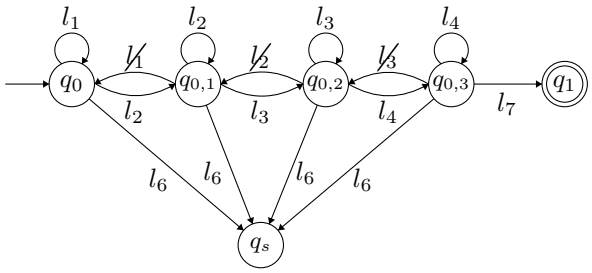

Fig. 6: Enlarged DFA of overtake scenario.

The enlarged DFA $\hat{\mathscr{F}}$ obtained from Algorithm 1 has a simulation relation to the original one $\mathscr{F}$. The definition of the simulation relation is given in Definition 4.

Definition 4. *Let $\mathscr{F} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$ be the DFA for a dynamical system. A DFA $\hat{\mathscr{F}} = (\hat{\mathcal{Q}}, \hat{\Sigma}, \hat{\delta}, \hat{q}_0, \hat{F})$ has a simulation relation to the DFA $\mathscr{F}$ if there exists a relation $\mathcal{R} \subseteq \mathcal{Q} \times \hat{\mathcal{Q}}$ such that*

1) $(q_0, \hat{q}_0) \in \mathcal{R}$
2) $\forall (q_i, \hat{q}_j) \in \mathcal{R}$, *if* $\hat{q}_j \in \hat{F}$, *then* $q_i \in F$
3) *for any* $(q_i, \hat{q}_j) \in \mathcal{R}$, *if* $\hat{q}_j \xrightarrow{l} \hat{q}'_j$, *then* $q_i \xrightarrow{l} q'_i$ *with* $(q'_i, \hat{q}'_j) \in \mathcal{R}$, *for some* $q'_i \in \mathcal{Q}$.

$\mathscr{F}$ and $\hat{\mathscr{F}}$ is in a simulation relation, denoted as $\mathscr{F} \succeq \hat{\mathscr{F}}$ if relation $\mathcal{R}$ for $(\mathscr{F}, \hat{\mathscr{F}})$ exists [3].

Given the original DFA $\mathscr{F}$ and enlarged DFA $\hat{\mathscr{F}}$, Algorithm 1 ensures that the simulation relation $\mathscr{F} \succeq \hat{\mathscr{F}}$ is satisfied as shown in Theorem 1.

Theorem 1. *Let $\mathscr{F} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$ be the original DFA and $\hat{\mathscr{F}} = (\hat{\mathcal{Q}}, \hat{\Sigma}, \hat{\delta}, \hat{q}_0, \hat{F})$ be the enlarged DFA obtained from Algorithm 1. There exists a relation $\mathcal{R}$ such that $\mathscr{F}$ and $\hat{\mathscr{F}}$ are in simulation relation $\mathscr{F} \succeq \hat{\mathscr{F}}$.*

Ensuring the simulation relation $\mathscr{F} \succeq \hat{\mathscr{F}}$ implies that if a word accepted by enlarged DFA $\hat{\mathscr{F}}$, it is also accepted by the original DFA $\mathscr{F}$ as

$$w \vDash \hat{\mathscr{F}} \rightarrow w \vDash \mathscr{F}, \tag{32}$$

where w is the word. Since the specification φ remains the same in both DFAs, the satisfaction probability $P(w \vDash \varphi)$ will be the same. Notably, while Algorithm 1 increases the number of states, it reduces the number of accepted words.

Proof. Let the original DFA $\mathscr{F}$ have $N+1$ states, excluding the sink and the final state. In this case, the state q_0 is considered the initial state. Before the construction of the enlarged DFA, the relation $\mathcal{R}$ is initialized as

$$\mathcal{R}_{orig} \coloneqq \{(q_0, \hat{q}_0), \ldots, (q_N, \hat{q}_N), (q_s, \hat{q}_s), (q_f, \hat{q}_f)\} \subseteq \mathcal{R}, \tag{33}$$

where q_n, q_s, q_f are the n-th, sink and final state. Since constructing an enlarged DFA adds states, the relation of the initial and final states stays the same, thus satisfying the first and second conditions.

If the connected polytope path exists in the state, states are added between the current $\hat{q}_n$ and the next state $\hat{q}_{n+1}$. All additional states have a relation with the current state q_n. The relation $\mathcal{R}$ is updated as

$$\mathcal{R} \coloneqq \{(q_n, \hat{q}_{n,1}), (q_n, \hat{q}_{n,2}), \ldots, (q_n, \hat{q}_{n,A})\} \cup \mathcal{R}_{orig}, \tag{34}$$

where the $\hat{q}_{n,i}$ is the i-th additional state to the n-th original state. The subscript A indicates the number of additional states calculated from the number of transitions. The transitions between any combinations from the current state $\hat{q}_n$ and the additional states $\hat{q}_{n,i}$, including two of the same state, are initiated by actions of self-loops in current state x_n. Therefore, the transitions satisfy

$$\begin{aligned} &\text{if } \hat{q}_{n,i} \xrightarrow{l} \hat{q}_{n,j} \text{ then } q_n \xrightarrow{l} q_n \\ &\text{with } (q_n, \hat{q}_{n,j}) \in \mathcal{R} \qquad \forall i, j \in [1, A]. \end{aligned} \tag{35}$$

For incoming and outgoing transitions from and to the next state $\hat{q}_{n+1}$, it satisfies

$$\begin{aligned} &\text{if } \hat{q}_{n,i} \xrightarrow{l} \hat{q}_{n+1} \text{ then } q_n \xrightarrow{l} q_{n+1} \text{ with } (q_{n+1}, \hat{q}_{n+1}) \in \mathcal{R} \\ &\text{if } \hat{q}_{n+1} \xrightarrow{l} \hat{q}_{n,i} \text{ then } q_{n+1} \xrightarrow{l} q_n \text{ with } (q_n, \hat{q}_{n,i}) \in \mathcal{R} \\ &\forall i \in [1, A]. \end{aligned} \tag{36}$$

Furthermore, the additional states have transitions to the sink state, which also satisfies

$$\begin{aligned}&\text{if } \hat{q}_{n,i} \xrightarrow{l} \hat{q}_s \text{ then } q_n \xrightarrow{l} q_s \text{ with } (q_s, \hat{q}_s) \in \mathcal{R} \\ &\forall i \in [1, A].\end{aligned} \tag{37}$$

All the equations of transitions Eq. (35), (36) and (37) satisfy the third condition for simulation relation when there is a path.

In case that the path does not exist, additional nodes are added to the next state. Therefore, the additional relation created in this step is

$$\begin{aligned}\mathcal{R} \coloneqq \{&(q_{n+1}, \hat{q}_{n+1,1}), (q_{n+1}, \hat{q}_{n+1,2}), \dots, \\ &(q_{n+1}, \hat{q}_{n+1,B})\} \cup \mathcal{R}_{orig},\end{aligned} \tag{38}$$

where the subscript B is the number of next states. The transition from current state $\hat{q}_n$ to next state $\hat{q}_{n+1,i}$ satisfies

$$\begin{aligned}&\text{if } \hat{q}_n \xrightarrow{l} \hat{q}_{n+1,i} \text{ then } q_n \xrightarrow{l} q_{n+1} \\ &\text{with } (q_{n+1}, \hat{q}_{n+1,i}) \in \mathcal{R} \qquad \forall i \in [1, B],\end{aligned} \tag{39}$$

since all the actions that initiate the outgoing transition are connected with an "or" operator $\vee$. Furthermore, the transitions of the self-loop at the next state satisfy

$$\begin{aligned}&\text{if } \hat{q}_{n+1,i} \xrightarrow{l} \hat{q}_{n+1,i} \text{ then } q_{n+1} \xrightarrow{l} q_{n+1} \\ &\text{with } (q_{n+1}, \hat{q}_{n+1,i}) \in \mathcal{R} \qquad \forall i \in [1, B].\end{aligned} \tag{40}$$

In case there is a sink connected to the addition states, the transitions satisfy

$$\begin{aligned}&\text{if } \hat{q}_{n+1,i} \xrightarrow{l} \hat{q}_s \text{ then } q_n \xrightarrow{l} q_s \\ &\text{with } (q_s, \hat{q}_s) \in \mathcal{R} \qquad \forall i \in [1, B].\end{aligned} \tag{41}$$

All of the equations of transitions eq. (39), (40) and (41) satisfy the third condition for simulation relation when there is not a path.

The conditions for simulation relation have been met for both of the cases. Therefore, the enlarged DFA obtained by the Algorithm 1 has a simulation relation to the original DFA $\mathscr{F} \geq \hat{\mathscr{F}}$. Furthermore, the two DFA cannot be in bisimulation relation. The transition condition is based on the intersection of polytopes being a facet. Since the transition conditions are restrictive, the bisimulation conditions cannot be met. □

Furthermore, the enlarged DFA obtained from Algorithm 1 follows Theorem 2. The pruning of transitions can be beneficial for simplifying the structure of the DFA.

Theorem 2. *Let $\mathscr{F}$ be the original DFA and $\hat{\mathscr{F}}$ be the enlarged DFA obtained from Algorithm 1. The elimination of transitions between states in $\hat{\mathscr{F}}$ does not alter the simulation relation between the two $\mathscr{F} \geq \hat{\mathscr{F}}$.*

D. DFA Patterns and LTL formulation

A subgraph of the enlarged DFA is selected to simulate and model check. In this subgraph, there are multiple reach-avoid problems (Appendix A). The letters that define the self-loop and the transition to the next state are a conjunction of letters in each decoupled system as

$$l \coloneqq l_1 \wedge l_2 \wedge \cdots \wedge l_N, \tag{42}$$

where N represents the number of decoupled dynamical systems. Fig. 7 shows the reach-avoid pattern in the 2D space. Transitioning to the next state is possible by changing the

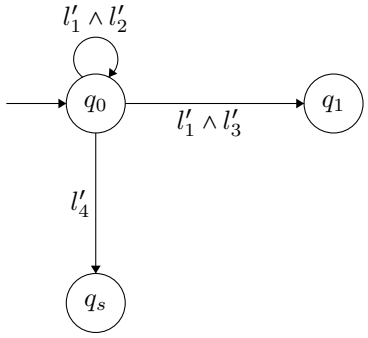

Fig. 7: Example of a reach-avoid pattern in the subgraph of enlarged DFA.

letter. The specifications are simplified to $\square(l'_1)$ for unchanged specifications and $l'_2 \mathsf{U} l'_3$ for changed specifications. However, having an "always" operator has multiple problems. Including the "always" operator prevents it from being a syntactically co-safe LTL specification. Furthermore, the satisfaction probability yielded would only contain zeros and ones for invariance problems. Time-bounding the specification can solve these problems. The scLTL specification can be obtained by

$$\begin{aligned}\square^{\leq I}(l'_1) &\equiv \bigwedge_{i=0}^{I} \bigcirc^{i}(l'_1) \\ l'_2 \mathsf{U}^{\leq I} l'_3 &\equiv \bigvee_{i=1}^{I} ((\bigwedge_{j=0}^{i-1} \bigcirc^{j}(l'_2)) \wedge \bigcirc^{i}(l'_3)),\end{aligned} \tag{43}$$

where I represents the time-bound. The time-bound is a parameter that needs to be tuned for optimal control. The decoupled scLTL formula is needed to run the modular solution.

With the inspection of the enlarged DFA of the overtake scenario (Fig. 6), there are four distinct reach-avoid patterns from q_0 to $q_{0,1}$, $q_{0,1}$ to $q_{0,2}$, $q_{0,2}$ to $q_{0,3}$, and $q_{0,3}$ to q_1. For example, the first subproblem can be represented as a reach-avoid problem where the agent wants to reach l_2 but avoids 1_6. Furthermore, when it is decoupled, the letters can be written as

$$\begin{aligned}&l_1 : p_1 \wedge (\neg p_2 \wedge p_3) \\ &l_2 : \neg p_1 \wedge (\neg p_2 \wedge p_3) \\ &l_6 : p_1 \wedge (\neg p_2 \wedge \neg p_3),\end{aligned} \tag{44}$$

where p_1 is for the y direction and combination of p_2 and p_3 represents the y direction. Based on the decoupled letter, the DFA of the first pattern is shown in Fig. 8. The decoupled specification can be written as $\square^{\leq I}(\neg p_2 \wedge p_3)$ for the x direction and $p_1 \mathsf{U}^{\leq I} \neg p_1$ for the y direction.

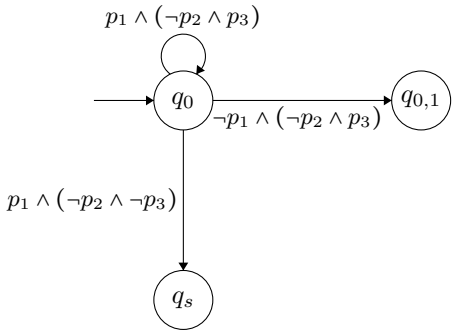

Fig. 8: First reach-avoid pattern of overtake scenario.

E. Value Iteration

In model checking, the value iteration is done to obtain the value function (satisfaction probability) [23], [24]. The value function V_k^{μ} can be considered the satisfaction of specification at time step k. This interpretation comes from the Bellman equation used for each iteration update looking at one time step ahead. In the modular solution, the value iteration is applied to each pattern to obtain the value function and find the optimal time-bound. The value iteration of a pattern shown in Fig. 7 is written as

$$\begin{aligned} V_{i,k+1}^{\mu_{i,k}}(s,q) &= T(V_{i,k}^{\mu_{i,k}})(s,q) \\ &= \mathbb{E}[\mathbb{1}_{(l_1' \wedge l_3')}(q^+) + \mathbb{1}_{(l_1' \wedge l_2')}(q^+) V_{i,k}^{\mu_{i,k}}(s^+,q^+)], \end{aligned} \tag{45}$$

where the $l_1' \wedge l_3'$ represents the area to reach and $l_1' \wedge l_2'$ represents the safe areas. Define indicator functions to see if the specifications are satisfied as

$$\begin{aligned} \mathbb{1}_{l_1'}(s^+) &= \begin{cases} 1 \text{ if } s^+ \in S(l_1) \\ 0 \text{ else} \end{cases} \\ \mathbb{1}_{\neg l_1'}(s^+) &= \begin{cases} 1 \text{ if } s^+ \in S(\neg l_1) \\ 0 \text{ else.} \end{cases} \end{aligned} \tag{46}$$

The indicators for l_2' and l_3' are defined similarly. Based on these indicator functions, the value iteration can be rewritten as

$$\begin{aligned} V_{i,k+1}^{\mu_{i,k}}(s,q) &= \mathbb{E}[\mathbb{1}_{(l_1' \wedge l_3')}(q^+) + \mathbb{1}_{(l_1' \wedge l_2')}(q^+) V_{i,k}^{\mu_{i,k}}(s^+,q^+)] \\ &= \mathbb{E}[\mathbb{1}_{l_1'}(s^+)\mathbb{1}_{l_3'}(s^+) \\ &\quad + \mathbb{1}_{l_1'}(s^+)\mathbb{1}_{l_2'}(s^+) V_{i,k}^{\mu_{i,k}}(s^+,q^+)]. \end{aligned} \tag{47}$$

Based on the value function obtained by value iteration, policy for the i-th pattern can be obtained by

$$\mu_i(s) = \arg\max_{q'} V_{i,k}^{\mu_{i,k}}(s,q'), \tag{48}$$

where q' represents the DFA states, excluding the sink and final states. The optimal time-bound is calculated from the value function for all state combinations. However, the value iteration is done for all the patterns with the decoupled dynamical system. The value function merging the decoupled systems can be calculated using Theorem 3.

Theorem 3. *Let there be N decoupled systems with separate value functions. The value function of j-th decoupled system following a certain policy μ_j is denoted as $V_{k,j}^{\mu_j}$. The value function merging the decoupled systems V_k^{μ} following a policy μ is calculated by taking the tensor product (denoted as $\otimes$) as*

$$V_k^{\mu} = V_{k,1}^{\mu_1} \otimes V_{k,2}^{\mu_2} \cdots \otimes V_{k,N}^{\mu_N}, \tag{49}$$

where μ is the policy obtained from merging each policy μ_j.

Proof. Let there be N decoupled systems with separate value functions. The value function of j-th decoupled system following a certain policy μ_j is denoted as $V_{k,j}^{\mu_j}$. When utilizing the definition of the value function in model checking (satisfaction probability), the value function merging the decoupled systems $V_k^{\mu}(s,q)$ can be written as

$$V_k^{\mu}(s,q) \xrightarrow{k \to \infty} \mathbb{P}(\pi_0 \vDash \varphi | s,q), \tag{50}$$

where the w_k is the word defined by the policy. Since the specification can be decoupled into smaller dimensions φ_j connected with an "and" operator, the value function can be written as

$$\mathbb{P}(\pi_0 \vDash \varphi | s,q) = \mathbb{P}(\pi_0 \vDash (\varphi_1 \wedge \varphi_1 \wedge \cdots \wedge \varphi_N) | s,q), \tag{51}$$

where φ_i is the specification for i-th decoupled system when there are N decoupled systems. Furthermore, using the independence of each of the decoupled systems, the value function can be written as

$$\begin{aligned} &\mathbb{P}(\pi_0 \vDash (\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_N) | s,q) \\ &= \mathbb{P}((\pi_{t,1} \wedge \pi_{t,2} \wedge \cdots \wedge \pi_{t,N}) \vDash (\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_N) | s,q) \\ &= \mathbb{P}(\pi_{t,1} \vDash \varphi_1 | s_1) \mathbb{P}(\pi_{t,2} \vDash \varphi_2 | s_2) \ldots \mathbb{P}(\pi_{t,N} \vDash \varphi_N | s_N) \\ &= V_{k,1}^{\mu_1}(s_1) V_{k,2}^{\mu_2}(s_2) \ldots V_{k,N}^{\mu_N}(s_N), \end{aligned} \tag{52}$$

where the $w_{k,j}$ and s_j are the word defined by policy $\mu_{k,j}$ and decoupled states. The value function $V_{k,j}^{\mu_j}(s_j)$ is a vector of satisfaction probability for every state in the respective systems. From this, it can be concluded that Eq.(52) is in the form of the tensor product. Therefore, the value function can be obtained by

$$V_k^{\mu} = V_{k,1}^{\mu_1} \otimes V_{k,2}^{\mu_2} \cdots \otimes V_{k,N}^{\mu_N}. \tag{53}$$

□

F. Parameter optimization

The value function V_k^{μ}, a tensor whose dimensionality depends on the number of decoupled systems, presents a computational difficulty. Direct calculation of the value function for all possible states and time-bounds makes it impractical due to memory restrictions. To address this, machine learning, specifically a multi-layer perceptron (MLP), is employed to approximate the optimal time-bound I^*, shown as

$$I^* = \arg\max_k V_k^{\mu}, \tag{54}$$

for each of the subproblems. The MLP is trained on a dataset of known optimal time-bounds for different combinations of

decoupled states. The MLP aims to accurately predict the optimal time-bound for states with stochasticity.

The MLP consists of three layers of fully connected nodes. The input layer receives the state combination and the values over different time-bounds. The number of nodes in the hidden layer is a parameter that can be tuned for efficiency and accuracy. Furthermore, a rectified linear unit (ReLU) is used for the nonlinear activation function. The output is a regression layer that predicts the optimal time-bound. The structure and model selection are based on the efficiency of the implementation. The mean-squared-error (MSE) is used in model training and for the evaluation of the model, and the evaluation of the model is based on accuracy calculated as

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of total predictions}}. \tag{55}$$

Based on the predicted time-bound, the dependencies between the subproblems are incorporated by updating the target polytope and specification. Let I^* be the predicted time-bound for i-th subproblem. Based on the optimal time-bound I^*, the target polytope can be written as

$$P_{t,i}(p_t) = \{s | V^{\mu}_{I^*,i+1}(s,q) \geq thres\}, \tag{56}$$

where the polytope for a certain atomic proposition p_t is a set of states s where the value function for $i+1$-th subproblem is larger than a certain threshold. Based on this atomic proposition p_t, the updated specification can be written as $\neg p_t \mathrm{U}^{\leq I^*} p_t$ for reach-avoid problem and $\Box^{\leq I^*}(p_t)$ for invariance problem.

IV. Parameter Tuning and Dynamics Assessment

As presented in the previous section, the modular solution provides a framework for efficient model checking systems and synthesizing a controller that maximizes specification satisfaction. However, the solution's effectiveness and accuracy depend on several factors, including the choice of parameters and system dynamics. The effects of these factors are investigated in this section using 1D and 2D car parking as explained in Huijgevoort and Haesaert [16].

A. Selection of Simulation Relation Parameter

The (ε, δ)-stochastic simulation relation establishes a link between the dynamical system M and the finite state abstraction $\hat{M}$ [16]. This simulation relation can be invaluable to verification and control design. ε represents the maximum deviation of output states between a system and its abstraction. It quantifies how closely the states need to be. On the other hand, δ represents the maximum deviation in the transition probability of a system and its abstraction. It quantifies the closeness of transition behavior. The selection of the parameters is crucial for the accuracy of the abstraction. The simulation relation defines how a state on M relates to $\hat{M}$ as

$$\mathcal{R} \coloneqq \{(\hat{x}, x) \in \hat{\mathbb{X}} \times \mathbb{X} \mid ||x - \hat{x}||_E \leq \varepsilon\}, \tag{57}$$

where $||x||_E = \sqrt{x^\top E x}$ [14]. When the E matrix is an identity matrix, this equation states that the nearest center point of the grid is the state of the finite state abstraction. The ε acts as a threshold for defining the nearest point. Based on this simulation relation $\mathcal{R}$, the requirement for the ε can be written as

$$\varepsilon > \sqrt{g^\top E g}. \tag{58}$$

The grid size $g \in \mathbb{R}^{n\times 1}$ is a parameter for finite state abstraction of a n-dimensional system. Rounding up the value obtained from $\sqrt{g^\top E g}$ is the minimal ε value.

In the case of the car park, the ε value is rounding up the grid size since it is one-dimensional. Since the $g = 0.1$, the minimal ε value is $\varepsilon = 0.1$. This ε value is only theoretical since it does not guarantee the highest initial satisfaction probability. The Gaussian process (Appendix B) is used to approximate the effect of ε on satisfaction. The result of applying the GP to the satisfaction probability with different ε values is shown in Fig. 9. The satisfaction probability

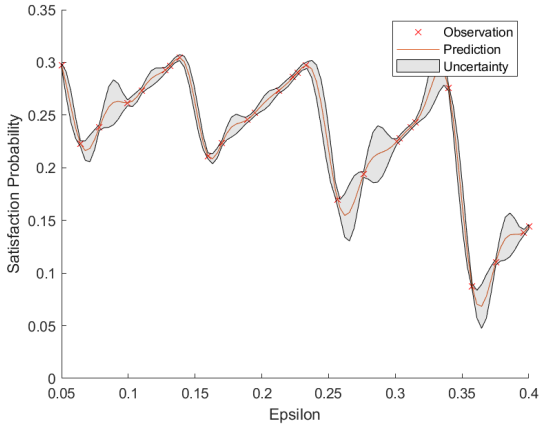

Fig. 9: GP on satisfaction probability with varying ε.

decreases further away from the target, and the sample point is the furthest away, making the satisfaction probability low. The satisfaction probability at minimal value $\varepsilon = 0.1$ is 0.25. The highest satisfaction probability is achieved when $\hat{\varepsilon} = 0.34$. From the obtained results, the minimal ε value and the actual $\hat{\varepsilon}$ value that maximizes the satisfaction probability differ.

The grid size of the finite state abstraction can be used to find the minimal ε value. The selection of the ε value is crucial since it affects the satisfaction of a specification. However, choosing a minimal ε value does make a strict relation but does not guarantee the highest satisfaction. The optimal ε can be selected using Gaussian processes to approximate the effects of ε.

B. Evaluation of Different Dynamics and Stochasticity

The dynamical system defines how the vehicle moves in the environment. The vehicle dynamics and stochasticity of the states are closely connected to the probability of satisfying the specifications. Therefore, the effects of dynamics and stochasticity on satisfaction probability will be further investigated. The vehicle dynamics are defined in the A matrix of the state space. The range of the A matrix is set as $[0.8, 2.0]$ for 1D

and $[0.7, 1.0]$ for 2D, considering the range of parking spaces. The evaluation result is shown in Fig. 10 and Fig. 11.

The maximum satisfaction probability is centered around $A = 1$ for 1D and $A = 0.85$ for 2D. In the case of large values of A, the vehicle movement between time steps would be large and miss the parking regions. On the other hand, if the A value is small, movement over time steps is small, and the target cannot be reached. For low A values, the satisfaction probability is considerably low compared to high A values in the 1D case. While having a large A value can be compensated by smaller input values, having a small A value cannot be compensated by the input since the input is bounded. Furthermore, the parking space is small, considering the size of the gridding, which makes it difficult to reach if the A value is large. The adjustment in the size of the parking space could affect the probability of satisfaction.

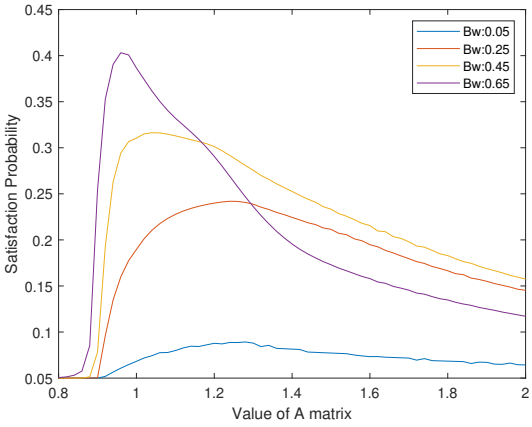

Fig. 10: Evaluation of different dynamics (1D case).

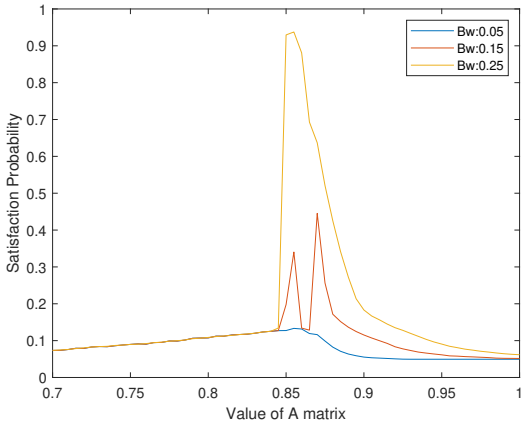

Fig. 11: Evaluation of different dynamics (2D case).

The stochasticity affected the peak value and the point of highest satisfaction probability. The peak satisfaction probability increases, and the value of the peak shifts to the left with stochasticity. The stochasticity broadens the range of possible states, making states closer to the target being included. Since the states closer to the target are included, the satisfaction of reaching the target increases.

This investigation demonstrates the relationship between the dynamics and stochasticity of the system and the satisfaction probability. There is a specific value of A that maximizes the satisfaction probability. This result suggests a trade-off in the state evolution over the time step, which can lead to overshooting and struggling to reach the target. Furthermore, stochasticity affects peak satisfaction by broadening the possible states, including ones closer to the target.

V. Numerical Experiments

This section introduces the application of modular solutions to two use cases: double integrator and single integrator overtake. These experiments are used to validate the efficiency and effectiveness of the modular solution when applied to complex dynamical systems using SySCoRe toolbox [14].

A. Implementation

It is first tested on double integrator overtake described in Sec. II to evaluate the performance of the modular solution. The grid size of the abstraction is set to $g = (60, 30)$ for relative position and velocity, and the time interval is set to $\Delta t = 0.5$. Computational time and memory usage of the modular solution and the original 4D model checking are compared. The reduction rate is used as a metric to see the efficiency of the modular solution, which is calculated as

$$\text{Reduction Rate} = \frac{\text{Performance of 4D model checking}}{\text{Performance of Modular Solution}}, \quad (59)$$

where the performance is the computational time and memory usage. Furthermore, Monte Carlo simulation (Appendix C) is applied since the state evolution is stochastic. The controller implementation is done 1000 times to obtain the expected states. Next, the modular solution is similarly applied to a single integrator overtake. The single integrator dynamical system for the overtake is shown as

$$\begin{cases} \Delta x_{t+1} = \Delta x_t + \Delta v_{x,t} \Delta t + w_{1,t} \\ \Delta y_{t+1} = \Delta y_t + \Delta v_{y,t} \Delta t + w_{2,t}, \end{cases} \quad (60)$$

where the states are relative position $(\Delta x, \Delta y)$ and inputs are relative velocity $(\Delta v_x, \Delta v_y)$ similar to one introduced in Sec. II. The specification, DFA, and polytopes remain the same as the double integrator overtake as introduced in Sec. II. The grid for abstraction is set to $g = 60$ and $\Delta t = 0.5$.

B. Modular Solution on Double Integrator Overtake

The computation time and memory usage of the 4D and modularized model checking are shown in Table II. Both the calculation time and memory usage were significantly reduced. This reduction comes from the decomposition of DFA and decoupling of the dynamical system addressing the curse of dimensionality. The modularized solution incorporates both aspects to make model checking significantly efficient.

First, the controller is independently implemented with a randomly selected time-bound of 4 time steps. This result is

TABLE II: Calculation time and memory usage comparison of two model checking methods without controller optimization.

	Time [sec]	Memory [MB]
4D Model Checking	1410	12496.27
Modularized Solution	2.64	80.75
Reduction Rate	534.09	154.75

shown in Fig. 12 and 13. The figures show the mean relative position and velocity in the x and y directions obtained from 1000 simulations for the four subproblems in the overtake scenario. For the position (Fig. 12), the standard deviation of the y direction is shown as an example. The collision area is avoided for each pattern, and the specification is satisfied based on the analysis of relative positions. Although there are tendencies to decrease the velocity, the overall relative velocity remains in the range where specification can be satisfied.

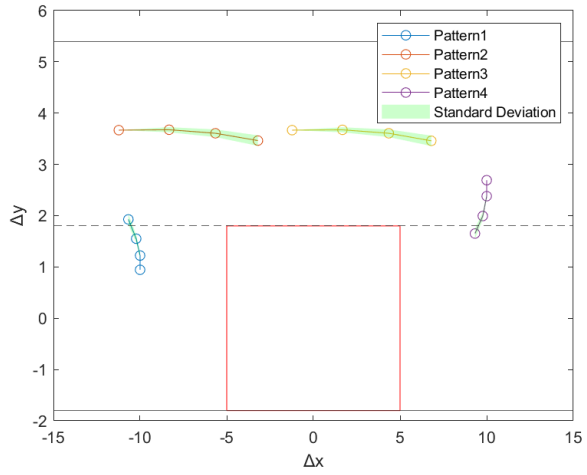

Fig. 12: Relative position of an unoptimized 4D controller.

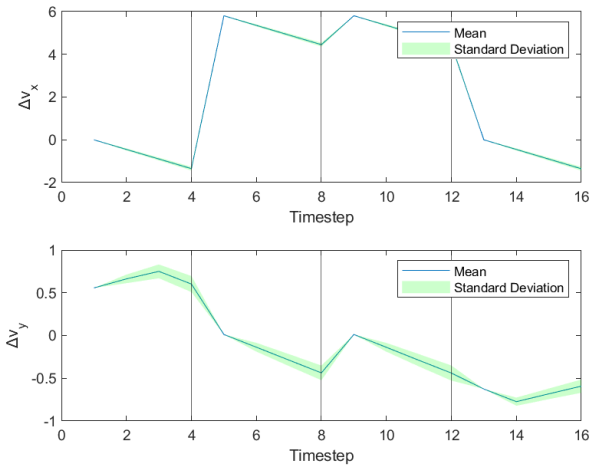

Fig. 13: Relative velocity of an unoptimized 4D controller.

Next, the controllers with dependency are implemented. While the model checking finishes without any problems, the controller implementation fails. Specifically, the specification is satisfied for the first subproblem, but all subproblems after it violates the specifications. The controller implementation failure happened since the ending state of the first subproblem was the state where satisfaction with the next subproblem was low; thus, it could not simulate the other subproblems fully.

C. Modular Solution on Single Integrator Overtake

The results of controller implementation with dependencies are shown in Fig. 14.

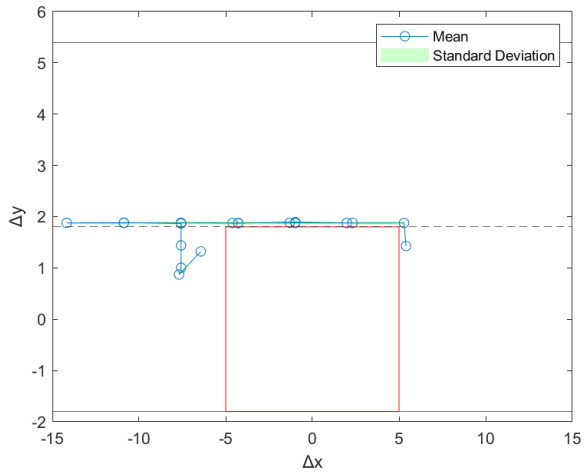

Fig. 14: Relative position of an unoptimized 2D controller.

The figure demonstrates the successful execution of the overtake. The time-bound predicted by the machine learning model was $(4, 7, 7, 1)$ time steps for each of the subproblems. As the vehicle trajectory shows, unnecessary movements were getting the vehicle further from the target polytopes. The unnecessary movements are due to the absence of a cost function to penalize the time step it takes to satisfy specifications inside SySCoRe. Additionally, the trajectory is close to the center line of the lanes. This trajectory comes from two aspects of model checking: finite state abstraction, where the gridding is too coarse, and polytopes defined by letters are too large. Another explanation for the trajectory is from the specification and simulation since the simulation ends when the specification is satisfied.

VI. Discussion

This paper introduces a modular solution, a novel compositional verification method for an approximated automaton, to address the state space explosion problem in model checking. By decomposing an approximated automaton into subproblems and integrating a machine learning model for time-bound prediction, the method demonstrates promising results in model checking and controller synthesis for single integrator overtake. The modular solution leverages decoupled dynamics and independent model checking to solve the global problem efficiently.

However, the current method faces limitations in handling double integrator dynamics due to low satisfaction in the second subproblem. The large target polytope of the first problem includes states with low satisfaction for the second subproblem, which contributes to the failure in controller implementation. Furthermore, the selected case study utilizes one case of the state-splitting algorithm, leaving the other unverified. Another problem is that the hyperparameters in the machine learning model have not been optimized.

Future research could investigate applications to higher-dimensional dynamical systems by optimizing target polytopes and specifications for a predicted time-bound. The optimization of target polytopes and specifications can avoid the low satisfactory ending states for the first problem, allowing full controller implementation. Conducting additional case studies with various scenarios could verify the whole modular solution. The hyperparameters and structure of the machine learning model can be looked into to improve the efficiency of the modular solution further.

VII. Conclusion

In conclusion, this paper introduces a promising modular solution for reducing the state space explosion problem in model checking. While the current approach shows potential for efficiency improvement and low dimensional dynamical systems, further research and refinements are needed to tackle the limitations and realize its potential as an efficient approach to model checking and controller synthesis in various applications.

Acknowledgment

I would like to express my gratitude to my thesis supervisors, Dr. Sofie Haesaert, Zengjie Zheng, and Shuhao Qi for their guidance, support, and feedback throughout the research. Their guidance has been crucial in assisting my academic growth and successful completion of my thesis.

References

[1] R. ltd and Markets, *Future of autonomous systems - focus on autonomous navigation software market - a global and regional analysis: Focus on application, sector, platform, software technology, and country - analysis and forecast, 2023-2033*, 2023.

[2] J. Chen, J. Sun, and G. Wang, "From unmanned systems to autonomous intelligent systems," *Engineering*, vol. 12, pp. 16–19, May 2022.

[3] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of model checking*. MIT Press, The, 2016.

[4] A. Paigwar, E. Baranov, A. Renzaglia, C. Laugier, and A. Legay, "Probabilistic collision risk estimation for autonomous driving: Validation via statistical model checking," *2020 IEEE Intelligent Vehicles Symposium (IV)*, Oct. 2020.

[5] J. Ma, X. Che, Y. Li, and E. M.-K. Lai, "Traffic scenarios for automated vehicle testing: A review of description languages and systems," *Machines*, vol. 9, no. 12, p. 342, 2021.

[6] G. Bagschik, T. Menzel, and M. Maurer, "Ontology based scene creation for the development of automated vehicles," *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[7] F. Nejati, A. A. Ghani, N. K. Yap, and A. B. Jafaar, "Handling state space explosion in component-based software verification: A review," *IEEE Access*, vol. 9, pp. 77 526–77 544, 2021.

[8] P. Godefroid and S. Khurshid, "Exploring very large state spaces using genetic algorithms," *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 266–280, 2002.

[9] U. Stern and D. L. Dill, "Improved probabilistic verification by hash compaction," *Lecture Notes in Computer Science*, pp. 206–224, 1995.

[10] J. R. Burch, E. M. Clarke, and D. Long, *Symbolic Model Checking with Partitioned Transition Relations*, Oct. 1991.

[11] S. Berezin, S. Campos, and E. M. Clarke, *Compositional reasoning in model checking*, Feb. 1998.

[12] C. Belta, B. Yordanov, and E. A. Gol, *Studies in Systems, Decision and Control 89*. Springer, 2017.

[13] T. Sarkar, A. Rakhlin, and M. A. Dahleh, "Finite time lti system identification," *Machine Learning Research*, 2021.

[14] B. Van Huijgevoort, O. Schön, S. Soudjani, and S. Haesaert, "Syscore: Synthesis via stochastic coupling relations," *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*, May 2023.

[15] M. Kloetzer and C. Mahulea, "Multi-robot path planning for syntactically co-safe ltl specifications," *2016 13th International Workshop on Discrete Event Systems (WODES)*, May 2016.

[16] B. C. van Huijgevoort and S. Haesaert, "Similarity quantification for linear stochastic systems: A coupling compensator approach," *Automatica*, vol. 144, p. 110 476, Oct. 2022.

[17] G. M. Ziegler, *Lectures on Polytopes*. Springer, 1995.

[18] M. Wiering and M. v. Otterlo, *Reinforcement learning: State-of-the-art*. 2012.

[19] A. S. Poznyak, *Advanced mathematical tools for automatic control engineers: Stochastic techniques*. Elsevier Science, 2009.

[20] D. Blackwell, "On stationary policies," *Journal of the Royal Statistical Society. Series A (General)*, vol. 133, 1970.

[21] P.-J. Meyer and D. V. Dimarogonas, "Hierarchical decomposition of ltl synthesis problem for nonlinear control systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 11, pp. 4676–4683, Nov. 2019.

[22] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Decomposition of finite ltl specifications for efficient multi-agent planning," *Distributed Autonomous Robotic Systems*, pp. 253–267, 2018.

[23] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, no. 11, pp. 2724–2734, Nov. 2008.

[24] S. Summers and J. Lygeros, "Verification of discrete time stochastic hybrid systems: A stochastic reach-avoid decision problem," *Automatica*, vol. 46, no. 12, pp. 1951–1961, Dec. 2010.

[25] B. Landry, M. Chen, S. Hemley, and M. Pavone, "Reach-avoid problems via sum-or-squares optimization and dynamic programming," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2018.

[26] P. I. Frazier, *A tutorial on bayesian optimization*, 2018. arXiv: 1807.02811 [stat.ML].

[27] E. Brochu, V. M. Cora, and N. d. Freitas, *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning*, Dec. 2010.

[28] H. Yu, T. Xie, S. Paszczynski, and B. M. Wilamowski, "Advantages of radial basis function networks for dynamic system design," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 12, pp. 5438–5450, 2011.

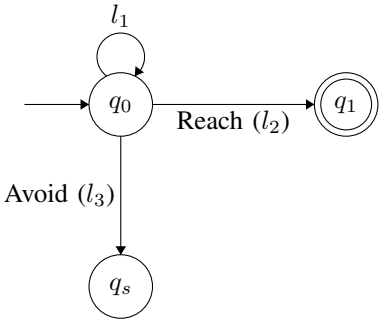

Fig. 15: Reach-avoid problem.

APPENDIX

A. Patterns of Subproblems

The decoupled subproblem can be classified into the reach-avoid problem and the invariance problem. Reach-avoid problem is a common problem in control, where the agent tries to reach a specific target without entering the avoiding regions [25]. For these problems, the specifications can be written as $l_1 \mathsf{U} l_2$ where l_1, l_2, and l_3 is the letter that initiates the transition for self-loop, reach, and avoid. Fig. 15 shows the deterministic finite automaton of a reach-avoid problem. An invariance problem is a control problem where an agent tries to keep within a target. For these problems, the specification can be written as $\square(l_1)$.

B. Bayesian Optimization

Bayesian optimization is a machine learning technique to directly search the minima or maxima of the objective function using Bayes' Theorem, written as

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}, \tag{61}$$

where A and B is the observation or event and where $\mathbb{P}$ is the probability $\mathbb{P}(B) : B \to [0,1]$ that maps event to a value [26]. The nomenclature for these probabilities is posterior for $\mathbb{P}(A|B)$, prior for $\mathbb{P}(A)$, likelihood for $\mathbb{P}(B|A)$, and marginal for $\mathbb{P}(B)$. The Bayesian inference utilizes Bayes' theorem to calculate the posterior based on prior beliefs. In the case of regression, where the objective is to learn the function, the prior is the prior belief of the function. The Bayesian inference is done when given a new point sampled from the function to calculate the updated belief about the function. The Bayesian optimization is useful for expensive sampling where the number of samples is limited or acquiring one sample is time-consuming [27]. Bayesian optimization is well-suited for situations where the data has random noise. One of the most used surrogate models for Bayesian optimization is called Gaussian Process (GP).

GP is a machine learning method, and one of the ways to utilize it is through regression. The GP is a Bayesian optimization method and a probability-based non-parametric model. Therefore, modeling a function and prediction can be done with uncertainty.

GP assumes that the data points are sampled from a multivariate Gaussian distribution. The main idea is to learn the distribution of the function by updating the mean and covariance function using a GP model denoted as GP, as shown in

$$f(z) \sim GP(\mu(z), \kappa(z, z')), \tag{62}$$

where the mean $\mu(z)$ and covariance $\kappa(z, z')$ is expressed as

$$\begin{aligned} \mu(z) &= \mathbb{E}[f(z)] \\ \kappa(z, z') &= \mathbb{E}[(f(z) - \mu(z))(f(z') - \mu(z'))]. \end{aligned} \tag{63}$$

Given a set of observed data, Bayesian inference calculates the posterior. The posterior can be taken as the prediction with the uncertainty of the corresponding observed data. The covariance function is also called a kernel, where a matrix displays the relation between the points in the function to be approximated. The kernel can be seen as a prior of the function, and it will be updated with Bayesian inference. The most used kernel functions are radial basis functions (squared exponential kernel), and given the points z_1 and z_2, it shows how similar these points are. The radial basis function has the advantage of simplicity, generalizability, and robustness [28].

In the GP, the data is randomly generated, and the inference only needs to be done once. However, this does not guarantee the lowest uncertainty in the learned function. Active learning is commonly used to address this problem. Active learning in machine learning is a method that interacts with the dynamical system and gathers and labels the data. In active learning, the new data points are decided by taking the highest uncertainty (variance) as

$$z^* = \max_z \kappa(z, z). \tag{64}$$

The new points are selected and generated iteratively to reduce the uncertainty in the function to approximate. This iteration process reduces the time to learn the function as the new points lower the uncertainty in the highest areas.

C. Monte Carlo Simulation

A Monte Carlo simulation is a mathematical method to approximate the possible outcomes of uncertain events. The outcome is approximated by the law of large numbers, as shown as

$$\frac{1}{N}\sum_{i=1}^{N} x_i \xrightarrow{N\to\infty} \mathbb{E}(X), \tag{65}$$

where the average of N realizations x_i approaches the expected value of the random variable X as the number of realization N goes to infinity. The Monte Carlo simulation performs N realizations of states with stochasticity. The average of the N realizations is taken to approximate the expected states.