Formal synthesis of controllers for safety-critical autonomous systems: Developments and challenges

Xiang Yin a, Bingzhao Gao b, Xiao Yu c,∗

a Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China
b School of Automotive Studies, Tongji University, Shanghai 201804, China
c Institute of Artificial Intelligence, Xiamen University, Xiamen 361005, China

A R T I C L E I N F O

Keywords:
Autonomous systems
Safety critical
Formal methods
Correct-by-construction synthesis

A B S T R A C T

In recent years, formal methods have been extensively used in the design of autonomous systems. By employing mathematically rigorous techniques, formal methods can provide fully automated reasoning processes with provable safety guarantees for complex dynamic systems with intricate interactions between continuous dynamics and discrete logics. This paper provides a comprehensive review of formal controller synthesis techniques for safety-critical autonomous systems. Specifically, we categorize the formal control synthesis problem based on diverse system models, encompassing deterministic, non-deterministic, and stochastic, and various formal safety-critical specifications involving logic, real-time, and real-valued domains. The review covers fundamental formal control synthesis techniques, including abstraction-based approaches and abstraction-free methods. We explore the integration of data-driven synthesis approaches in formal control synthesis. Furthermore, we review formal techniques tailored for multi-agent systems (MAS), with a specific focus on various approaches to address the scalability challenges in large-scale systems. Finally, we discuss some recent trends and highlight research challenges in this area.

1. Introduction

With the rapid advancement of information technology, control technology and artificial intelligence, autonomous systems are now extensively utilized in various areas of our society, including industrial manufacturing systems (Fragapane, Ivanov, Peron, Sgarbossa, & Strandhagen, 2022; Theorin et al., 2017), intelligent transportation systems (Chen et al., 2022; Schwarting, Alonso-Mora, & Rus, 2018), and healthcare in daily life (Attanasio, Scaglioni, De Momi, Fiorini, & Valdastri, 2021; Navarro et al., 2021). Notably, these autonomous systems often operate within complex and dynamic environments with uncertainties and potential adversarial components. A defining characteristic of autonomous systems is their comprehensive integration of perception and decision-making modules, accompanied by substantial computational capabilities. Consequently, a central challenge in autonomous systems lies in how to effectively make decisions online in order to effectively leverage information acquisition and computational resources, ensuring accurate and timely commands in the face of the complexities of their operational environments.

A fundamental characteristic of autonomous systems is that they are typically classified as safety-critical systems (Dawson, Gao, & Fan, 2023; Wolf & Serpanos, 2017; Xiao, Cassandras, & Bela, 2023), which implies that any malfunction in their functionalities could result in significant catastrophes. For instance, in autonomous vehicles, safety is consistently the foremost concern, as any error in their control software has the potential to endanger the lives of passengers (Selvaraj, Ahrendt, & Fabian, 2022; Zhao, Yurtsever, Paulson, & Rizzoni, 2022). Similarly, in manufacturing systems, production robots must operate in a correct sequence to prevent collisions or buffer overflows (Chryssolouris, 2013). Another example is scenario of urban search and rescue robots, where they need to ensure both the safety of the robot itself and the individuals being rescued. This requires to navigate to a desired region promptly while avoiding unknown obstacles (Delmerico et al., 2019).

Nevertheless, synthesizing controllers for autonomous systems poses a significant challenge due to the complex dynamics of the systems and the intricate design objectives involved. In particular, autonomous systems are also characterized as cyber-physical systems, featuring continuous dynamics at the physical level and embedded control logics at the software level (Alur, 2015; Lee & Seshia, 2016; Platzer, 2018). Consequently, ensuring safety for this class of hybrid dynamic systems necessitates addressing design concerns at both the low-level, such as collision avoidance, and the high-level, ensuring logical correctness so that tasks are executed in the correct order.

∗ Corresponding author.
E-mail addresses: yinxiang@sjtu.edu.cn (X. Yin), gaobz@tongji.edu.cn (B. Gao), xiaoyu@xmu.edu.cn (X. Yu).

https://doi.org/10.1016/j.arcontrol.2024.100940
Received 10 December 2023; Accepted 20 February 2024
Available online 12 March 2024
1367-5788/© 2024 Elsevier Ltd. All rights reserved.
The hybrid nature of autonomous systems and the safety specifications not only make control strategies designed susceptible to errors but also may contribute to a prolonged and time-consuming design phase. Designing autonomous systems based solely on human experience may not be sufficient to address the inherent complexity challenges. To ensure safety and streamline the design process, one may seek that control synthesis procedures possess the following two key properties:

- **Correctness Guarantees:** Upon the generation of the control law by the synthesizer, formal guarantees regarding its correctness can be deduced through rigorous mathematical tools. This eliminates the need for manual examination and redesign of controllers, thereby not only reducing the duration of the design phase but also enhancing the reliability of the controller.

Generally speaking, formal methods refer to the use of mathematically rigorous techniques to ensure the correctness of complex systems. It was originally developed within the field of computer science in the context of software verification and synthesis, offering a powerful framework to fulfill the requirements aforementioned (Clarke & Wing, 1996; Manna & Pnueli, 2012). Notably, by leveraging rigorous mathematical techniques to formalize both the system model $\Sigma$ and the desired specifications $\phi$, one can ensure provably correct guarantees for reasoning processes and results. Within the context of formal methods, two fundamental problems emerge: verification problems and synthesis problems. In the verification problem, the task is to check whether a given (closed-loop) system model satisfies some specific desired specifications. Conversely, in the synthesis problem, the system model is open and reactive, and the objective is to synthesize a program to determine inputs online so that the given specification can be enforced. The formal synthesis problem is inherently related to the control problem, since the program is essentially a feedback controller. This design paradigm of formal synthesis is also commonly referred to as “correct-by-construction” control synthesis.

Over the past decades, there has been a significant increase in attention towards applying formal methods to control synthesis of autonomous systems (Belta & Sadraoddini, 2019; Belta, Yordanov, & Gol, 2017; Kress-Gazit, Lahijanian, & Raman, 2018; Li, Serlin, Yang, & Belta, 2019; Luckcuck, Farrell, Dixon, & Fisher, 2019; Mehdipour, Althoff, Tebbens, & Belta, 2023). A major catalyst for this trend is the capability of formal methods to provide a unified approach for describing and managing heterogeneous desired safety-critical design objectives, encompassing both low-level dynamics and high-level logics. For instance, in autonomous robots, a fundamental task involves the “reach-avoid task” (Fridovich-Keil et al., 2020; Summers & Lygeros, 2010) requiring that the robot should visit a target region “Goal” while avoiding unsafe regions “Bad” during its operations. This task can be succinctly captured by a linear temporal logic (LTL) formula (Baier & Katoen, 2008), such as $\varphi = F\text{Goal} \land G\text{-Bad}$, where $F$ denotes “eventually” and $G$ denotes “always”. Moreover, LTL specifications empower users to articulate more intricate safety-critical requirements for autonomous systems, such as controlling a target region infinitely often or executing various sub-tasks in a specified order.

This paper aims to offer a thorough survey of the basic methods, recent advances, and the state-of-the-art in the application of formal control synthesis techniques to safety-critical engineering cyber-physical systems, with a particular emphasis on autonomous systems such as teams of autonomous robots. Particularly, in this paper, we adopt a unified approach by treating safety requirements as the satisfaction of a logical formula $\varphi$. As discussed earlier, this formulation offers a unified method that bridges the gap between the satisfaction of physical constraints and the correctness of high-level logics. In summary, the primary focus is on addressing the formal control synthesis problems of various forms within the following context.

Given an autonomous system $\Sigma$ operating in an uncertain environment and a desired specification formula $\varphi$, synthesize a feedback control $C$ such that the closed-loop system $\Sigma_C$ satisfies the specification $\varphi$ with a certain level of formal guarantees.

The structure of this paper is shown in Fig. 1. In Section 2, we first classify formal control synthesis problems for autonomous systems. We also introduce some basic specification languages widely used in safety-critical formal control synthesis. In Section 3, we discuss how to synthesize controllers when systems are abstracted as symbolic models. Section 4 focuses on how to synthesize controllers directly based on the continuous dynamics of the system without discretizing the state spaces. When the system model is unknown to the user, data-driven synthesis approaches are introduced in Section 5. Section 6 extends control synthesis techniques to the case of multi-agent systems (MAS), where local agents may be coupled with each other by networks. We particularly focus on how to synthesize controllers for large-scale systems in a scalable manner and how to address new safety-critical requirements that arise in multi-agent systems. Finally, we conclude the paper in Section 7 by discussing more recent trends and research challenges formal control synthesis for autonomous systems.

## 2. Formal control synthesis problems

In this section, we categorize the formal control synthesis problems considered in this paper. Specifically, based on whether the system model is deterministic, non-deterministic, or probabilistic, the underlying synthesis problems are classified as planning problems, reactive synthesis problems, and probabilistic synthesis problems. Additionally, we review some commonly used formal specification languages for autonomous systems.

### 2.1. Formal system models

In this works, we follow the formalism by explicitly considering the dynamics of the autonomous system in an environment as a plant model. Roughly speaking, a system model is a tuple $\Sigma = (X, U, f)$, where $X$ is the state space, $U$ is the action space, and $f$ is the transition relation (Tabuada, 2009). In general, $X$ and $U$ can be uncountable infinite sets such as real numbers when the system is continuous. When they are countable sets, the system $\Sigma$ is referred to as a symbolic system. In numerous applications, a system $\Sigma$ is also augmented with a labeling function that specifies the atomic properties holding at each system state. Then, depending on the structure of the transition relation, formal models can be further categorized as follows.

**Deterministic Models:** A system is categorized as deterministic when the transition relation conforms to the form $f : X \times U \rightarrow X$. In essence, such a system executes control commands perfectly without encountering any disturbances. Consequently, for these ideal systems, a feedback control strategy is unnecessary, and one can simply design an open-loop plan for execution. For continuous systems, it can be the standard linear or nonlinear systems without disturbances. For symbolic systems, deterministic finite state automata or labeled transition systems represent the most conventional models. In the cases where the system exhibits countable but infinite states, Petri nets (Giua & Silva, 2018; Murata, 1989; Reisig, 2012) or pushdown automatas (Griffin, 2008; Schmuck, Schneider, Raisch, & Nestmann, 2016) offer suitable tools for capturing unbounded state spaces.
Non-Deterministic Models: A system is categorized as non-deterministic when the transition relation conforms to the form $f : X \times U \to 2^X$. For continuous systems, this can be differential inclusion systems with set-value mappings (Aubin & Cellina, 2012). In the context of symbolic systems, non-deterministic transition systems or finite-state automata with uncontrollable events are common representations (Cassandras & Lafortune, 2008). This modeling approach is typically employed to represent environments characterized by uncertainty, disturbances, or adversarial elements, particularly in a zero-sum context. In this non-deterministic setting, for a given pair of state and control input, the system may move non-deterministically to a set of possible states; which specific state to go depends on the choice of the environment.

Note that the physical interpretation of transition non-determinism varies depending on the specific problem scenarios. In a basic case, non-deterministic transitions may capture operation in a perturbable environment, where the actuator cannot execute a command perfectly due to unmodeled uncertainty. Additionally, non-deterministic transitions can represent the presence of an uncontrollable agent; for instance, a robot operating in a workspace with freely moving passengers. Moreover, non-determinism can capture unknown environments, where a state may have multiple possible properties before the agent actually visits it. The mathematical model of non-deterministic systems provides a unified framework to represent these diverse physical scenarios.

Stochastic Models: It is worth noting that, in non-deterministic models, there is no a priori knowledge regarding the likelihood of each possible transition. By incorporating this probabilistic priori information into the model, we obtain a stochastic model characterized by a transition relation of the form $f : X \times U \to \text{dis}(X)$, where $\text{dis}(X)$ is a probabilistic distribution on the state space $X$. This model is commonly denoted as the Markov decision process (Kumar & Varaiya, 2015; Puterman, 2014). Note that, for both non-deterministic or stochastic models, feedback control strategies are strictly necessary as the controller needs to adjust its plan online based on the behavior of the environment.

Multi-Player Models: The three classes of models aforementioned are widely used in capturing the robust operation of a single (team of) system in environments with uncertainties. Note that, although non-deterministic transitions are allowed in these models, the environment itself lacks a distinct objective. In certain scenarios, an additional player, distinct from the controlled system, may be introduced, each with its own specific objective. This objective may not be entirely adversarial to that of the controller. For instance, the input set can be partitioned as $U = U_1 \cup U_2$, and the transition function can be formulated as $f : X \times U_1 \times U_2 \to X$ for the case concurrent decision-making (De Alfaro, Henzinger, & Kupferman, 2007) or $f : U(X \times U) \to X$ for the case turn-based decision-making (Brenquier et al., 2016).

Stochasticity can also be integrated into multi-player models, leading to stochastic games (Křetínský, Rammeantu, Slivinskiy, & Weininger, 2022; Zhang, Yang, & Bazar, 2021).

Throughout this paper, our focus will be on reviewing the formal synthesis for the first three classes of systems. Specifically, we assume that all decision-makers are cooperative against the uncertain environment sharing the same overall objective. Multi-player decision-making in the context of non-zero-sum games is beyond the scope of this survey.

2.2. Formal specification languages

Given a system model $\Sigma$, whether in an open-loop configuration or closed-loop under control, each individual execution of the system generates a (finite or infinite) sequence of states (with labels if needed), which is referred to as a trace. The aggregate of all possible traces generated by $\Sigma$ is called its language, denoted as $L(\Sigma)$. Essentially, formal specifications serve the purpose of characterizing whether the language of $\Sigma$ is a “good” language.

It is important to note that, in general, a specification needs to evaluate $L(\Sigma)$ as a whole rather than individually evaluating each trace in it. A particular but notably powerful case is that of linear-time specifications, where the satisfaction of the specification can be evaluated by inspecting the satisfaction of each individual trace. For instance, “reach-avoid” stands as the most widely used linear-time specification. In the domain of safety critical control of autonomous systems, the following linear-time specifications are widely employed.

Linear Temporal Logic (LTL) is one of the most widely used formal specifications for describing the safety and correctness of autonomous systems (Pnueli, 1977). Specifically, an LTL formula $\psi$ is evaluated over infinite traces. In addition to standard Boolean operators, LTL allows temporal operators such that “always” $\text{G}$, “next” $\text{X}$, “eventually” $\text{F}$ and “until” $\text{U}$. Therefore, it supports complex safety-critical requirements such that “should not do task B before task A is finished”. A notable feature of LTL is that, for an arbitrary LTL formula $\psi$, it can be accepted by...
a non-deterministic Büchi automaton or a deterministic Rabin/parity automaton. Therefore, automata-theoretical approaches can be applied to model-checking and control synthesis problems. There are also many useful variants of LTL formulae such as co-safe LTL (scLTL) (Kupferman & Vardi, 2001), LTL over finite traces (LTLF) (De Giacomo & Vardi, 2013) and Truncated LTL (TLTL) (Li, Vasile, & Belta, 2017). Notably, scLTL formulae are widely used in describing tasks that can be satisfied within a finite horizon. Specifically, scLTL requires that negation operator can only be applied in front of atomic proposition; therefore, temporal operator “always” is not allowed. The main property of scLTL is that any infinite trace satisfying the formula has a “good” prefix and can be accepted by deterministic finite-state automata.

Note that the applications of LTL are restricted to traces that are purely logical and contain no real-time information. However, in real-world autonomous systems, the traces generated by the systems consist of real-valued signals in the time domain. Therefore, it becomes imperative to further extend LTL to encompass both continuous state space and real-time considerations. Metric Temporal Logic (MTL) stands out to further extend LTL to encompass both continuous state space and real-time considerations. Metric Temporal Logic (MTL) stands out as a prominent extension of LTL by introducing constraints on temporal operators with intervals of real numbers (Kooymans, 1990; Thati & Rosu, 2005). For instance, $F_{\leq 5} \psi$ denotes that the task $\psi$ can be satisfied within 4 to 5 time units from the present moment. Signal Temporal Logic (STL) (Fainekos & Pappas, 2009; Zhong, Jordan, & Provost, 2021), one can quantitatively evaluate the satisfaction degree of the task rather than receiving a Boolean answer.

The previously mentioned LTL, MTL, and STL, along with their variants, all belong to the category of linear-time properties. However, many applications require branching-time properties to capture the desired specifications. For example, one might necessitate that at each instant, the robot has an exit path to a safe region, but it is not necessarily obliged to execute it. This requirement cannot be captured by linear-time specifications but can be expressed using CTL* with temporal operators and path quantifiers, which is an extension of Computation Tree Logic (CTL) (Emerson & Halpern, 1986).

In the context of supervisory control of discrete-event systems, the well-adopted non-blockingness condition (Wonham, Cai, et al., 2019; Yin & Laforetice, 2015a, 2018) essentially represents a branching-time requirement. This condition can be captured by a CTL* formula such as $\text{AGFGoal}$ (Ehlers, Laforetice, Tripakis, & Vardi, 2017), where $A$ and $E$ are path quantifiers “for all” and “exists”, respectively.

2.3. Formal control synthesis problems

As discussed, the control synthesis problem and the corresponding formal guarantees vary for different types of system models. Here, we categorize the formal control synthesis problem into the following three categories.

Planning Problem: For a deterministic system model, employing a feedback control strategy is unnecessary. Instead, it suffices to identify an open-loop input sequence under which a path $\rho$ with finite or infinite states (or state labels) can be generated by system $\Sigma$. For planning problems, the consideration involves a linear-time property specification $\varphi$, and the corresponding formal guarantee is expressed as $\rho \models \varphi$.

Reactive Control Synthesis Problem: For a non-deterministic system model $\Sigma$, the controller $C$ is a feedback strategy that issues the next control input based on the observation history. In general, memory is required to realize the controller $C$, and the resulting closed-loop system $\Sigma_C$ can be considered as a synchronized product between the system model and the controller model. In this case, the trace generated by the system is not a single one but a language, and the specific trace generated online depends on the choice of the non-deterministic environments. The corresponding formal guarantee is expressed as $\Sigma_C \models \varphi$, meaning that the language generated by $\Sigma_C$ satisfies the specification. Particularly, when $\varphi$ is a linear-time specification, the formal guarantee can be equivalently expressed as $\forall \rho \in L(\Sigma_C) : \rho \models \varphi$. This is essentially a robust condition meaning that the system is always safe no matter what the environment does.

Probabilistic Synthesis Problem: When the system model $\Sigma$ incorporates probabilistic information in transitions, it becomes possible to discuss the likelihood of each trace generated. For instance, the system may not always satisfy the specification, but the probability of failure is very small. A typical formal guarantee for probabilistic synthesis is expressed as $P(\Sigma_C \not\models \varphi) \leq 1 - \epsilon$, where $P$ is a probability measure, and its event space encompasses the set of all possible languages. Finally, we remark that, for all these three types of formal synthesis problems, beyond the qualitative satisfaction of formal specifications, numerous quantitative cost criteria can also be imposed to evaluate the performances of the controller, which may further lead to multi-objective synthesis problems and constrained optimization problems.

3. Abstraction-based formal control synthesis

In this section, we focus on discussing formal control synthesis techniques based on symbolic system models. Although symbolic models are inherently limited by discrete state spaces, techniques associated with symbolic systems continue to play fundamental roles in formal control synthesis for several reasons.

- First, a considerable number of safety-critical systems inherently exhibit symbolic characteristics and it is sufficient enough to consider symbolic models at the task planning level. A notable example of this case is the autonomous robot with embedded controllers, where the control software operates within a symbolic framework (Kress-Gazit et al., 2018). Another example is the manufacturing process, where discrete states represents the configurations of the machines and complex sequences of events are involved to drive the evolution of configurations (Theorin et al., 2017).
- Second, although many more complex systems involve continuous state-spaces and time-driven dynamics, they can be abstracted as symbolic system, either precisely or approximately. This issue will be discussed in this section. Therefore, synthesis techniques can still be applied to their abstractions, and the resulting symbolic controllers can be refined back to control the original systems (Tabuada, 2009).
- Finally, it should be noted that most of the widely used formal specifications can be expressed through symbolic models. For instance, regular languages find representation through finite state automata, while linear temporal logic specifications can be captured by nondeterministic Büchi automata. Consequently, symbolic models offer a unified framework for aligning the dynamic behavior of a system with the task processes under investigation.

Therefore, a comprehensive understanding of synthesis techniques for symbolic systems becomes the foundational step towards formal synthesis for more complex systems. Existing works on formal control synthesis with symbolic models are summarized in Table 1.

3.1. Construct symbolic models via formal abstractions

As mentioned earlier, the initial phase of abstraction-based formal synthesis involves constructing the symbolic model from the original dynamic system through abstraction. Various approaches exist for building formal abstractions, and the choice depends on the nature of the underlying dynamic systems. Here, we briefly outline some commonly used techniques.
The most straightforward approach is to partition the operational state space based on atomic properties of interest. In this method, connected regions with the same properties can be abstracted as a single symbolic state. This technique is typically employed in broad and complex environments where the dynamics of the agent are not a primary concern. In such scenarios, feasible plans at the physical level can readily be found to navigate the agent from one state to another. For general dynamic systems operating in polygonal environments, computationally efficient algorithms have been developed in Fainekos, Girard, Kress-Gazit, and Pappas (2009), Kloetzer and Belta (2008) and Tabuada and Pappas (2003) to automatically partition the state space into polygonal triangulations. Based on these triangulations, vector fields can be constructed to generate trajectories navigating between each abstract state (Belta, Isler, & Pappas, 2005).

Another approach is to discretize the state-space so such some formal relationship between the abstract system and the concrete system can be established. For example, for planning problems for deterministic systems, one can use the notion of (approximate) (bi-)simulation relations (Girard & Pappas, 2007; Girard, Pola, & Tabuada, 2008; Pola, Girard, & Tabuada, 2008; Zamani, Pola, Mazo, & Tabuada, 2011). For reactive control synthesis problems, the alternating simulation relation can be used to handle transition non-determinism for each control action (Alur, Henzinger, Kupferman, & Vardi, 1998; Huth (2005), Kamide and Kupferman, 2006, Qian and Lin (2020), Wagner (2013), Wu and Lin (2015, 2016), Yoo, Fish, and Sakkariefi (2013), Zhang, Wu, and Lin (2015)) and can be used to refine the abstract states. These relations have also been extended to the probabilistic setting (Lavaei, Fainekos, and Pappas (2014), Hata (2013), Chen, Wang, et al. (2023), Kurtz and Lin (2023), Li and Fu (2021), Montana, Liu, and Dodd (2023), Zhou, Chen, et al. (2024), and Dimarogonas (2020), Hahn et al. (2019), Hasanbeig et al. (2019), Hashimoto, Tsumagari, and Ushio (2022, 2022), Vorobeychik et al. (2019), and Wang (2014), and Topcu (2012), Wolff et al. (2013), Wolff, Topcu, and Murray (2012), Wolff et al. (2013), Xie, Yin, Li, and Zhang (2021), Zhao, Zhu, Li, and Yin (2021), and Dimarogonas (2020)).
& Zamani, 2022; Zamani, Abate, & Girard, 2015; Zamani, Esfahani, Majumdar, Abate, & Lygeros, 2014; Zhong, Lavaei, Zamani, Caccamo, & Marco, 2023). There are also many other approaches to construct formal abstractions, such as l-complete approximations (Moor, Raisch, & O’young, 2002; Schmuck & Raisch, 2014) and feedback refinement relations (Khaled, Zhang, & Zamani, 2023; Reissig, Weber, & Rungger, 2016; Ren & Dimarogonas, 2020).

### 3.2. Methods for path planning

For deterministic symbolic systems, the path planning problem aims to find a feasible sequence such that a given specification $\phi$ is satisfied. The primary method for addressing this type of problem is through an automata-theoretical approach (Vardi, 2005). Notably, for any given specification $\phi$, a satisfiable path in the plant represents an instance of violation of the formula $\neg \phi$. Consequently, by conducting model checking on the negation of the task over the plan model, a feasible plan can be readily identified. Essentially, this approach involves first establishing the synchronization product of the specification automaton and the plant model.

The automata-theoretical approach has been employed by most formal planning algorithms, either explicitly or implicitly. The basic approach involves building the entire product system all at once and then performing a graph search for feasible and optimal paths over the product space, depending on the accepting condition of the specification automaton and the numerical metric for optimality (Guo & Dimarogonas, 2015; Smith et al., 2011; Ulusoy, Smith, Ding, BELTA, & Rus, 2013). However, this approach may not scale well when the size of the plant model increases. To address this, many works use the strategy of incrementally constructing the product space online as the system evolves. A very popular planning algorithm in this category is the sampling-based approach such as Rapidly Exploring Random Tree (RRT). In this approach, a tree over the product space is constructed incrementally from the initial root until a vertex formulating a satisfaction instant is visited, and it turns out that this approach scales very well for systems with very large state-spaces (Kantaros & Zavlanos, 2018, 2020; Liu, Li, & Yin, 2023; Vasile et al., 2020). Note that the incremental planning approach not only provides computational efficiency, but can also be applied to online path planning problems where the entire system is unknown a priori. In the context of LTL path planning, another efficient approach is to fix the planning horizon a priori and to encode the planning problem as a mixed-integer linear program (Sahin, Nilsson, & Ozay, 2019).

### 3.3. Methods for reactive control synthesis

For non-deterministic systems, controllers need to react to the behavior of the environment dynamically online. The exploration of formal control synthesis commenced in the 1980s, with contributions from both the computer science and control engineering communities in the contexts of reactive synthesis for software (Clarke & Emerson, 1981; Pnueli & Rosner, 1989) and supervisory control of discrete event systems (Ramadge & Wonham, 1987, 1989; Wonham et al., 2019), respectively. In the context of reactive synthesis, researchers mainly focus on open systems operating within reactive environments. The primary objective was to ensure that the generated output responded accurately to environmental input, thereby satisfying specified properties in the overall input/output (I/O) sequence. In this setting, the possible behavior of the environment, as well as the dynamics of the agent, can be encoded as an environment formula $\psi_{\text{env}}$ and the overall synthesis specification is of the form $\psi_{\text{env}} \rightarrow \psi_{\text{task}}$ (Kress-Gazit, Fainekos, & Pappas, 2009; Liu et al., 2013; Majumdar, Pitnerman, & Schmuck, 2019; Maniatopoulos et al., 2016). On the other hand, in the context of supervisory control theory, an explicit plant model under control is considered with the behavior of environments being modeled by uncontrollable events. In this scenario, a supervisor was introduced to disable the occurrence of controllable events, thereby enforcing the desired requirements. Essentially, both of these settings address similar problems and can be approached with a more unified perspective in the context of zero-sum two-player games (Gradels, Thomas, & Wilke, 2003) with qualitative or quantitative specifications.

Note that a key feature of controllers synthesized through two-player games is their ability to maintain satisfaction of the formal task regardless of the actions of the environments. This formal guarantee essentially requires the synthesis procedure to conduct a global search to recursively eliminate unsafe states, leading to the identification of a control invariant (also referred to as a winning region) (Bernet, Janin, & Walukiewicz, 2002; Dallal, Neider, & Tabuada, 2016). When strict global feasibility guarantees are not necessary or the global plant model is time-varying or even unknown a priori, path planning algorithms can also be employed in a receding-horizon fashion to yield a reactive control (Grover et al., 2021; Kantaros et al., 2022, 2020; Li et al., 2021). However, such planning-based control may be myopic in the sense that it may result in a situation where no further feasible plan exists.

In reactive control synthesis, since the feedback controller relies on online information to make decisions, it often faces the challenge of partial observability. The formal synthesis of reactive controllers with partial-observability has been extensively studied in the context of both supervisory control of discrete event system (DES) (Ji, Yin, & Lafontaine, 2021; Yin & Lafontaine, 2015b) and games with imperfect information (Berthon, Rubert, Murano, Rubin, & Vardi, 2021; Chatterjee, Doyen, Henzinger, & Raskin, 2007; Ramasubramanian et al., 2020). In this partial-observation setting, the agent may not only need to synthesize its control strategy, but may also have to jointly design its information acquisition strategy to dynamically deploy sensors on/off line (Fu & Topcu, 2016; Li, Wei, Li, & Yin, 2023). Another feature of the synthesis problem compared to the planning problem is that one can further enforce branching-time specifications such as CTL specifications (Jiang & Kumar, 2006).

### 3.4. Methods for probabilistic control synthesis

When faced with transition probability information, the probabilistic control synthesis problem arises as a combination of the standard Markov decision process for optimality criteria and reactive control synthesis for formal specifications (Cai, Xiao, Li, & Kan, 2021; Fu & Topcu, 2015b). This naturally results in a multi-objective optimization problem, necessitating a tradeoff between optimality performance and the satisfaction probability of the task. For example, Abate, Prandini, Lygeros, and Sastry (2008) studies probabilistic reachability and safety synthesis for stochastic hybrid systems. In the context of MDPs, Ding et al. (2014) addresses the problem of minimizing a specific type of expected cost across all controllers while maximizing the satisfaction probability of an LTL specification. Similarly, Guo et al. (2023) and Guo and Zavlanos (2018b) deals with the stochastic optimization problem, imposing the constraint that the satisfaction probability of an LTL specification is greater than a given threshold. Probabilistic Computation Tree Logic (PCTL) is also widely employed as the formal specification in probabilistic synthesis for MDP (Labhijanian et al., 2011).

### 4. Abstraction-free formal control synthesis

The aforementioned works on formal control synthesis in the previous section are performed on symbolic system models at the abstracted level. Although working on symbolic models provides a unified way to handle heterogeneous dynamics and specification, it faces the main challenge of the curse of dimensionality. Specifically, the size of the symbolic model grows exponentially as the dimension of the system increases. Furthermore, for real-valued and real-time logics such STL, one has to work directly based on the original system since these logics usually do not have automata representations. In this section, we review formal control synthesis techniques that work on continuous dynamic systems directly without the discretization of state space. Existing works on formal control synthesis with continuous models are summarized in Table 2.
4.1. Synthesis via optimizations

Given a discrete-time dynamic system over a continuous state space of the form $x_{t+1} = f(x_t, u_t)$ and a specification formula $\phi$, the objective is to find a control input $u_t$ such that $x_0, x_N \models \phi$, where $x_0, x_N$ is the solution state trajectory under $u_0, u_{N-1}$. The basic approach for solving this problem is through optimizations (Belta & Sadraddini, 2019), such as Mixed Integer Linear Programs (MILP). This approach was initially proposed by Raman et al. (2014) and Raman et al. (2015) for control synthesis for STL specifications, where the control objective is to maximize robustness. The main idea is to encode both the dynamics of the system and the STL specifications as MILP constraints, allowing existing optimization tools to be leveraged for finding a feasible solution. Beyond STL specifications, optimization-based control synthesis has also been applied to other specification logics such as MTL (Kurtz et al. 2017, 2018), Hashimoto, Hashimoto, and Takai (2022), Kalagarla et al. (2021), Lindemann, Pappas, and Dimarogonas (2021), Lindemann, Qin, Deshmukh, and Pappas (2023), Lindemann, Rodionova, and Pappas (2022), Venkataraman, Aksaray, and Seller (2020), Yan, Xu, and Li (2019), Yang, Zou, Li, and Yang (2023), Zhao et al. (2022). These approaches are significantly more scalable compared to precise encodings. However, they are not a complete methods and cannot be applied to complex formulae.

4.2. Synthesis via model predictive control

When uncertainties are encountered, the system model becomes $x_{t+1} = f(x_t, u_t, w_t)$ with $w_t$ representing the disturbance from the environment that the controller should account for. To design feedback control strategies, optimization-based planning can be seamlessly integrated into the framework of Model Predictive Control (MPC). Specifically, at each instant, one solves an optimization problem to generate an open-loop plan and only applies the first control input in the sequence to the system. The controller then repeats this process in a receding-horizon fashion, ensuring that disturbances are considered within the feedback loop.

MPC-based reactive control synthesis was initially developed for STL specifications with finite abstractions (Gol et al., 2015; Ulusoy & Belta, 2014; Wongpiromsarn et al., 2012). In the context of control synthesis for STL specifications, the MPC-based approach was developed together with the optimization-based approach in Raman et al. (2014) and Raman et al. (2015). In Sadraddini and Belta (2015), starting from a worst-case perspective, a robust MPC approach was proposed to ensure feasibility under uncertainties. In Farahani et al. (2017, 2018), Yang, Zou, Li, and Yang (2023) and Yang, Zou, Li, Yin, and Jia (2024), the shrinking horizon strategy was applied for stochastic dynamic systems under chance constraints. Similar to the optimization-based planning problem, the key bottleneck for MPC-based synthesis is

### Table 2

<table>
<thead>
<tr>
<th>Planning</th>
<th>Reactive control</th>
<th>Probabilistic synthesis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LTL</strong></td>
<td><strong>Aydin Gol and Lazar (2013)</strong></td>
<td><strong>Cai, Aysi, Belta, and Vasile (2023)</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Cai, Aysi, Belta, and Vasile (2017)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Fanekes et al. (2009)</strong>, Gol and Belta (2014)**</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Gol, Ding, Lazar, and Belta (2014)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Gol et al. (2013)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Karlsin, Vasile, Humova, Karman, and Rus (2018)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Karlsin et al. (2018)</strong>, Klothzer and Belta (2007)**</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Kreuza-Galet et al. (2009)</strong>, Meyer and Dimarogonas (2019)**</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Oh, Cho, Choi, and Oh (2020)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Srinivasan and Coogan (2020)</strong>, Wolff and Murray (2016)**</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Wolff, Topcu, and Murray (2014)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Yu and Dimarogonas (2021a)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>MTL</strong></td>
<td><strong>Barbosa, Lindemann, Dimarogonas, and Sumova (2019)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Donzé and Maler (2010)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Hoxha, Dokhanchi, and Fainekos (2018)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Koymann (1990)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Maler and Nickovic (2004)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>STL</strong></td>
<td><strong>Belta and Sadraddini (2019)</strong></td>
<td><strong>Aksaray, Jones, Kong, Schwager, and Belta (2016)</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Belta and Sadraddini, and Belta (2022)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Ghasemi, and Sadraddini (2021)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Kurtz and Lin (2020)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Leung and Pavone (2022)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Lindemann and Dimarogonas (2018)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Mehdipour, Vasile, and Belta (2019)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Raman, Donzé, and Zamani (2022)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Raman et al. (2014)</strong>, Raman et al. (2015)**</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Sadraddini and Belta (2015)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Yang, Zou, Li, and Yang (2023)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Planning</strong></td>
<td><strong>Xinyue Yin et al.</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Xinyue Yin, and Lindemann (2023)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Reactive control</strong></td>
<td></td>
<td><strong>Yu and Lin (2020)</strong>, <em>Haghighi et al. (2019)</em>, Kurtz and Lin (2020), Mehdiipour et al. (2019) and Pant et al. (2017).**</td>
</tr>
<tr>
<td><strong>Probabilistic synthesis</strong></td>
<td></td>
<td><strong>Haghighi et al. (2019)</strong>, Kurtz and Lin (2020), Mehdiipour et al. (2019) and Pant et al. (2017).**</td>
</tr>
</tbody>
</table>
still its scalability when the horizon is very long. To address this issue, recently, Yu, Wang et al. (2023) proposed more scalable approaches to pre-compute feasible sets in an offline fashion, reducing the online computation burden.

4.3. Synthesis via control barrier functions

To address computation challenges in control synthesis for temporal logic tasks, Control Barrier Functions (CBF), originally developed for safety-critical control of dynamic systems, have recently been adopted. The core idea of CBF involves describing the safe set as a super-level set of a smooth function $h(x)$. By enforcing the forward invariance of the safe set, safety can be guaranteed under any possible disturbances. Interested readers are encouraged to explore Ames et al. (2019), Ames, Xu, Grizzle, and Tabuada (2016), Huang et al. (2024), Prajna and Jadbabaie (2004), Wieland and Allgöwer (2007), Wongpiromsarn, Topcu, and Lamperski (2015), Xiao and Belta (2019), Yang, Chen, Preciado, and Mangharam (2022) for more in-depth details and recent advances on this topic including theoretical properties of CBFs, computations of CBF-based level sets and ways to find potential CBPs.

In the context of temporal logic specification, the work by Lindemann and Dimarogonas (2018) pioneered the application of CBF to control synthesis for STL specifications. The key step involves approximating the satisfaction region of the STL formula as a safe region. By enforcing safety within this defined safe region, STL specifications can be reliably guaranteed. This approach has garnered significant attention and has seen various extensions (Charitidou & Dimarogonas, 2023; Xiao et al., 2021). In terms of system models, the approach has been extended to diverse scenarios, including multi-agent systems (Lindemann & Dimarogonas, 2019, 2020), continuous-time systems (Yang, Belta, & Tron, 2020), and systems with input constraints (Buyukkocak et al., 2022). Expanding beyond STL, CBF and its variants, such as control barrier certificates, have found application in LTL specifications (Srinivasan & Coogan, 2020) and event-based STL specification (Gundana & Kress-Gazit, 2021, 2022). The primary advantage of the CBF-based approach lies in its high scalability, as it avoids combinatorial searches for binary variables. However, it is worth noting that CBF-based approaches are generally tailored to a simpler class of STL tasks, and for more complex STL tasks, this approach may potentially yield no solution even when a feasible one exists.

4.4. Synthesis via sampling-based approaches

Another popular synthesis approach for continuous dynamics without abstraction is the sampling-based approach. Similar to its symbolic counterpart, this approach does not directly handle the system dynamics but rather takes samples over the continuous state-space according to the system dynamics. The foundational work for this approach was introduced by Karaman and Frazzoli (2009, 2012), where Rapidly Exploring Random Graphs (RRG) were utilized for deterministic musculus specifications. Subsequently, Vasile and Belta (2013), Vasile et al. (2020) and Yu and Dimarogonas (2021a) extended this approach to handle LTL specifications. To enhance the scalability of the sampling-based approach, Bhatia et al. (2016), He et al. (2015) and Luo et al. (2021) employed RRT with biased sampling, achieving state-of-the-art performance. Moreover, the sampling-based approach has been adapted for stochastic settings for LTL specifications (Oh et al., 2020).

5. Data-driven formal control synthesis

For both abstraction-based and abstraction-free approaches discussed in the previous section, having a previous dynamic model of the system is crucial for control synthesis. However, there are scenarios where either a system model is not available beforehand, or the system dynamics are too complex to handle. In such cases, the interaction with the system through physical or simulation tests to collect data becomes essential. Furthermore, data-driven approaches can also improve the efficiency and scalability of the standard model-based approaches. In this section, we delve into data-driven approaches for formal synthesis with correctness guarantees.

5.1. Data-driven formal abstractions

Since symbolic models play a key role in formal synthesis, an immediate question arises: can we build symbolic models directly from data so that subsequent synthesis techniques can be applied? Recent years have witnessed a growing body of works exploring this direction. In studies such as Devanport, Saoud, and Arcak (2021) and Lavaei and Frazzoli (2022), a Probably Approximately Correct (PAC) statistical framework was developed for constructing symbolic models directly from data. Here, the fidelity of the abstraction to the underlying system is guaranteed by a PAC bound. This PAC-based framework has also been extended to different classes of systems in Coppola, Peruffo, and Mazzoni (2023) and Peruffo and Mazzoni (2022). Cubuktepe, Jansen, Junges, Katoen, and Topcu (2020) and Lavaei, Soudjani, Frazzoli, and Zamani (2022) investigated how to construct interval MDPs from data, where the transition probabilities are bounded to intervals with high confidence. The key theoretical foundation behind these approaches is the scenario approach (Camps, Carè, & Garatti, 2021; Campi, Garatti, & Prandini, 2009), which establishes formal confidence bound connections between sampled data and the underlying constraints.

5.2. Data-driven safe controller synthesis

The scenario approach employed in data-driven formal abstractions can be further leveraged for synthesizing a safe controller directly. By utilizing control barrier certificates, the safety requirement of an unknown system can be formulated as a robust convex program, which can be further relaxed into a scenario convex program. Building on this idea, Nejati, Lavaei, Jagtap, Soudjani, and Zamani (2023), Salamati, Lavaei, Soudjani, and Zamani (2021) and Salamati and Zamani (2022) addressed the data-driven safe verification problem. This approach has been extended to control synthesis problems, enabling the generation of controllers with formal safety guarantees expressed in terms of confidence bounds (Chen, Shang, Huang, & Yin, 2023; Salamati, Lavaei, Soudjani, & Zamani, 2024). Finally, in addition to the scenario approach, another formal alternative for ensuring safety via data is to use direct data-driven methods without explicitly identifying the unknown model (van Huijgevoort et al., 2023; Zhong, Zamani, & Caccamo, 2022).

5.3. Formal synthesis via reinforcement learning

Another popular approach for synthesizing specification-enforcing controllers for unknown dynamic systems is through reinforcement learning (RL), a sampling-based approach based on reward signals (Sutton & Barto, 2018). While most existing RL algorithms focus on optimizing scalar reward signals, recent developments have extended RL to synthesize controllers for MDPs with unknown transition probabilities under formal specifications (Hahn et al., 2019). For instance, in the context of LTL specifications for finite MDPs, model-free RL algorithms have been devised to maximize the probability of satisfying formal tasks (Bozkurt et al., 2020; Cui et al., 2023; Hasanbeig et al., 2019; Oura et al., 2020). Tabular-based RL algorithms for finite MDPs have also been developed under STL specifications using both model-free algorithms (Aksaray et al., 2016; Kalagarla et al., 2021; Venkataraman et al., 2020) and model-based algorithms (Cohen, Serlin, Leaby, & Belta, 2023; Kapoor, Balakrishnan, & Deshmukh, 2020). Moreover, RL algorithms have been extended to general MDPs over continuous state spaces for various formal specifications. Deep RL techniques are then leveraged in these cases instead of cumbersome tabular controllers (Cai, Asli, Belta, & Vasile, 2023; Cai, Hasanbeig, Xiao, and Abate, & Kan, 2021).
In the multi-agent setup, however, all three categories of synthesis approaches discussed above face the same challenges. The first and foremost challenge is the curse of dimensionality issue. In theory, if the number of agents increases, the state and action spaces also grow exponentially for MAS. Therefore, more scalable approaches, such as top-down approaches, bottom-up approaches or structural approaches, are needed. Also, in many MAS, the information structures for perceptions and control are naturally distributed, further necessitating the use of local control strategies. Finally, there is also a need for new underlying specification languages to capture the interactive nature of MAS. This section will elaborate on recent progress made to address these challenges.

6.1. Top-down approaches for MAS synthesis

In the context of MAS formal synthesis, a top-down approach involves initiating the synthesis process from the perspective of individual agents, gradually building up to the overall system behavior. A commonly employed bottom-up technique is incremental controller synthesis (Hill & Tilbury, 2008; Ulusoy et al., 2014), which explores the independence between each local module. Another powerful approach for bottom-up synthesis is the compositional reasoning (Alur et al., 2018; Ghasemi et al., 2022; Kulkarni & Fu, 2018; Liu, Saoud, Jagtap, Pokkink, & Rooda, 2019; Schmidt & Breindl, 2010; Wong & Wonham, 1996).

Top-down approaches for MAS synthesis have also been explored in the context of abstraction-free synthesis for continuous systems under STL specifications. For example, Charitatidou and Dimarogonas (2021) presented an approach to decompose STL specifications into local agents using convex optimization techniques. In Leahy et al. (2022), Satisfiability Modulo Theories (SMT) was employed to decompose a fragment of STL specifications known as Capability Temporal Logic (CaTL). Furthermore, in Yu and Dimarogonas (2021b), a hierarchical control strategy was developed for uncertain discrete-time nonlinear systems with STL specifications.

6.2. Bottom-up approaches for MAS synthesis

In contrast to the top-down approach, a bottom-up approach involves synthesizing the overall global specification directly from the perspective of individual agents, sequentially solving the problem for each agent or group. This approach is also referred to as contract-based synthesis (Ghasemi et al., 2022) or interface-based synthesis (Hill, Cury, de Queiroz, Tilbury, & Lafortune, 2010; Leduc, Brandin, Lawford, & Wonham, 2005).

6.3. Structural properties for MAS synthesis

When dealing with a set of homogeneous agents operating in the same environment, inherent similarities among the agents provide structural properties that can potentially reduce synthesis complexity, particularly in terms of the number of agents involved. For instance, the concept of permutation symmetry was employed in Nilsson and Ozay (2019) for formal control synthesis of large-scale systems under counting constraints. When the dynamics of each local agent is represented as a symbolic model, the overall MAS can be more efficiently captured as a Petri net model without explicitly enumerating the product state space. In the context of multi-robot planning, several works have concentrated on exploring the structural properties of Petri nets to efficiently synthesize control strategies (Kloetzer & Mahulea, 2020; Lv et al., 2023).
6.4. Communication-aware synthesis in MAS

In the context of formal methods for multi-agents systems, a noteworthy feature distinguishing it from the single-agent setting is the presence of an information structure. Each agent may possess its unique information, shared through a communication network with its neighbors within a certain range. In this context, the synthesis task involves not only finding local control strategies for each agent but also planning their communication protocols. This scenario is termed as communication-aware synthesis, presenting additional challenges due to potential information asymmetry among agents. For example, in the context of multi-robot planning, a communication-aware planning framework was introduced for STL specifications (Liu et al., 2020). Assuming that agents can only communicate when they are close to each other, control synthesis algorithms have been developed for intermittent communication scenarios, catering to both LTL specifications (Guo & Zavlanos, 2018a; Kantaros et al., 2019) and MTL specifications (Wang et al., 2022; Xu et al., 2023).

6.5. MAS related formal specifications

Standard formal specifications such as LTL and STL were originally developed for single-agent systems, lacking an explicit consideration of the multi-agent context. In scenarios with a large number of agents, several new semantics for temporal logic specifications have been proposed to effectively capture features specific to multi-agent systems. For instance, in Sahin et al. (2019), Counting Linear Temporal Logic (cLTL) was introduced to express requirements related to the number of agents simultaneously performing a given task. Similarly, Cardona et al. (2023) extended MTL to the swarm robot setting by incorporating considerations for the maximum number of sub-swarms. Additionally, Swarm Signal Temporal Logic (SwarmSTL) was introduced in Yan and Julius (2022) and Yan et al. (2019) to address monitoring and control of MAS with real-valued trajectories. Furthermore, the research in Djeumou, Xu, and Topcu (2020) and Xu, Nettekoven, Julius, and Topcu (2019) investigated the inference and control synthesis problems for swarm MAS with spatial–temporal properties, employing the notion of Graph Temporal Logic (GTL).

7. Recent advances and challenges

The preceding sections have provided an overview of fundamental aspects in formal control synthesis for safety-critical autonomous systems. However, as autonomous systems operate in complex real-world environments, there are more additional design challenges to consider, in particular, with the rapid developments of network and AI technologies. In this concluding section, we delve into related research topics, recent advances and research challenges in the field.

7.1. Robustness of controllers

In formal control synthesis, most works that provide formal safety guarantees rely on a reliable system model. However, when the behavior of the agent deviates from the nominal model, the safety guarantee may no longer hold, or the performance may change significantly. Therefore, it is crucial to investigate the robustness of the synthesized controller (Majumdar, Render, & Tabuada, 2011; Meira-Góes, Kang, Lafortune, & Tripakis, 2023; Rungger & Tabuada, 2015; Zhang et al., 2023). For instance, the robust semantics of STL has provided a useful way to quantify the robustness of satisfaction when spatial values change. More recently, temporal robustness has gained attention capturing how robust the performance of the controller is when the agent deviates from the nominal trajectory in time (Rodionova et al., 2022a, 2022b; Sahin et al., 2019). This temporal robustness issue is particularly important for MAS, where agents often operate asynchronously without a global clock (Lindemann et al., 2022; Wang, Li, & Yin, 2024; Yu, Yin, & Lindemann, 2023). Additionally, in the formal synthesis of MAS, quantifying the robustness of the entire system concerning the failure of each agent is also an important topic (Huang et al., 2022, 2022; Kalluraya et al., 2023b; Zhao, Wang, & Yin, 2023).

7.2. Security-aware formal synthesis

Existing works on formal control synthesis have primarily focused on safety concerning physical dynamics or correctness regarding logical behaviors. However, with the development of networked control systems, autonomous systems naturally operate in information-rich environments where communications and information transmissions are unavoidable. Therefore, security has become an increasingly critical issue to consider in formal synthesis, as it is closely related to the safe operation of the system (Liu, Trivedi, Yin, & Zamani, 2022; Yin, Zamani, & Liu, 2020). Recent efforts have concentrated on synthesizing controllers that ensure the security of the information-flow generated by the system preventing sensitive information from leaking to the outside under the notions of opacity (Shi, He, Ma, Ran, & Yin, 2023; Xie et al., 2021; Yang, Yin, Li, & Zamani, 2020; Yu et al., 2022) and differential privacy (Chen, Leahy, Jones, & Hale, 2023; Jones, Leahy, & Hale, 2019; Xu et al., 2020). This information-flow security synthesis problem has also been addressed more recently within the general framework of controller synthesis for hyper-properties (Bonakdarpour & Finkbeiner, 2020; Bonnah, Nguyen, & Hoque, 2023; Clarkson & Schneider, 2010; Finkbeiner, Rabe, & Sánchez, 2015; Wang, Nalluri, & Pajic, 2020; Zhao, Li, & Yin, 2024). Additionally, many works are investigating the formal synthesis problem for controllers that remain robust against the attackers attempting to actively hack into the control or observation channels (Fu, 2022; Kulkarni, Han, Leslie, Kamhoua, & Fu, 2021; Ma & Cai, 2021, 2022; Niu, Fu, & Clark, 2020; Udupa et al., 2022; Yao, Li, & Yin, 2024; Yao, Yin, & Li, 2020).

7.3. Synthesis under unstructured environments

From an application perspective, although safety guarantees can be provided, many existing formal synthesis techniques are currently applicable only to structured and ideal environments such as indoor warehouses. Extending formal control synthesis to autonomous systems operating within complex and unstructured environments introduces significant challenges. The primary challenge lies in dealing with uncertain and unknown environments, necessitating the use of sophisticated perception and navigation tools. For instance, autonomous robots in unknown environments usually require to leverage SLAM techniques to build maps and plan concurrently (Cristofalo et al., 2017; Kalluraya et al., 2023b; Kantaros et al., 2020; Li et al., 2022; Tian et al., 2022; Zhou, Chen, et al., 2024; Zhou, Wang, et al., 2023). Due to the use of unverified perception modules, to ensure safety, one may further need to appropriately quantify perception errors in the presence of rich but uncertain information (Li, Yang, Jia, Zhuang, & Mitra, 2023; Păsăreanu et al., 2023; Yang, Pappas, Mangharam, & Lindemann, 2023).

7.4. AI-enabled formal synthesis

Finally, the development of artificial intelligence (AI) has significantly expanded the potential scope of formal control synthesis for autonomous systems. Deep neural networks, for example, enable autonomous systems equipped with powerful perception modules to classify information in unstructured environments. Moreover, recent developments in large-language models (LLM) offer a more feasible way...
to make high-level decisions in complex environments (Devlin, Chang, Lee, & Toutanova, 2018; Vaswani et al., 2017). Although AI techniques have significantly enhanced the capabilities of autonomous systems and typically scale very well, the fundamental challenge with AI-enabled approaches is the lack of formal guarantees. Recently, there has been a growing number of works aiming to develop frameworks that ensure formal guarantees with the context AI-enabled safety-critical control synthesis, ranging from formally certifying AI-based control or perception modules (Huang et al., 2020; Ivanov et al., 2021; Xiang, Tran, & Johnson, 2018) to quantify uncertainty for AI-based plans to ensure formal guarantees (Calinescu et al., 2022; Chen, Rosolia, Fan, Ames, & Murray, 2021; Crouse et al., 2023; Kumar et al., 2023; Lindemann, Cleaveland, Shim, & Pappas, 2023; Lindemann, Qin, Deshmukh, & Pappas, 2023; Qin, Xia, Zatsui, Fan, & Deshmukh, 2022; Ren et al., 2023; Wang et al., 2023). Also, AI techniques have also been applied to enable formal synthesis for high-dimensional dynamical systems (Bansal & Tomlin, 2021; Onken et al., 2022; Yang, Chen, Yin, & Mangharam, 2024). In general, there directions are still in their early stages and require more efforts.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (62173226, 62173283, 62061136004).

References

Cai, M., Hasanbeig, M., Xiao, S., Abate, A., & Kan, Z. (2021). Modular deep reinforce-

}


Xiao, W., & Belta, C. (2019). Control barrier functions for systems with high relative degree. *In IEEE conference on decision and control (pp. 474–479).*


Xu, Z., Vahidi, K., Hale, M. T., & Topcu, U. (2020). Differentially private controller synthesis with metric temporal logic specifications. *In American control conference (pp. 4745–4750).*


Yao, J., Yin, X., & Li, S. (2020). On attack mitigation in supervisory control systems: A tolerant control approach. *In IEEE conference on decision and control (pp. 4504–4510).*


Yoo, C., Fitch, R., & Sukkarieh, S. (2013). Provably-correct stochastic motion planning with safety constraints. *In IEEE international conference on robotics and automation (pp. 981–986).*


Yu, P., & Dimarogonas, D. V. (2021b). Hierarchical control for uncertain discrete-time nonlinear systems under signal temporal logic specifications. *In IEEE conference on decision and control (pp. 1450–1455).*


Yu, X., Yin, X., & Lindemann, L. (2023). Efficient STL control synthesis under asynchronous temporal robustness constraints. *In IEEE conference on decision and control (pp. 6841–6848).*


Zhang, Y., Lai, A., Lan, W., & Yu, X. (2023). Optimal path planning with opacity-preservation temporal logic specifications using bipartite synthesizers. *In IEEE conference on decision and control (pp. 7862–7867).*


