IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 31 January 2024, accepted 1 March 2024, date of publication 7 March 2024, date of current version 14 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3374329

RESEARCH ARTICLE

Abstraction-Based Safe Control With Alternating Simulation-Based Shields and Its Application to Mobile Robots

MASASHI MIZOGUCHI[1] AND TOSHIMITSU USHIO[2], (Member, IEEE)
[1]Research and Development Group, Hitachi Ltd., Hitachi, Ibaraki 319-1292, Japan
[2]Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka 560-8531, Japan

Corresponding author: Masashi Mizoguchi (masashi.mizoguchi.re@hitachi.com)

This work was supported by Hitachi Ltd., as a Joint Research with Osaka University.

ABSTRACT Abstraction-based formal synthesis with a symbolic control barrier function is useful for obtaining a finite-state safe controller for an infinite system with sporadic disturbances. In the case of multiple mobile robots sharing a common workspace, a controller ensuring arrival to destinations without any collisions is obtained with the symbolic control barrier function, despite the existence of unpredictable sporadic packet dropouts among robots. In the existing method, a local abstracted model of each robot is constructed, they are composed to obtain an abstracted model of the entire system, and unsafe transitions in it are eliminated with a symbolic control barrier function. However, a considerable computation time is required when the number of robots is large. To solve this problem, a shield called an *alternating simulation-based shield (AS-Shield)* was introduced into abstraction-based formal synthesis. As well as the existing method, a local abstracted model for each robot was constructed. Instead of the composition, a local controller was constructed for each robot, and a safe controller for the robot was obtained by attaching an AS-Shield. Because a composition to obtain the entire system is not necessary, the control inputs are computable, even though the number of robots is large. To confirm the validity of the proposed method, it was implemented using a robot simulator.

INDEX TERMS Alternating simulation, cyber-physical systems, formal methods, hybrid systems, mobile robots, shield synthesis.

I. INTRODUCTION

Safety is one of the most important requirements for a system consisting of multiple mobile robots sharing a common workspace such as warehouses, autonomous cars, automated construction machineries, and unmanned aerial vehicles because collisions between robots or between a robot and an obstacle must not occur. The collision avoidance method proposed in [1] and [2] is not robust to environmental changes. Thus, minimally invasive safe controllers have been considered, such that the control and safety specifications are considered separately. There are two approaches for minimally invasive safe controllers: a control barrier function-based approach and an optimization-based approach. In particular, control barrier function-based approaches have received increasing attention because they require less computational time than optimization-based approaches [3], [4], [5]. The control barrier function is a mapping from a state space to a set of real numbers and is used to design a controller enforcing set invariance that typically characterizes the safety of dynamical systems. Nagumo showed a necessary and sufficient condition for set invariance [6]. This condition was applied to design controllers that make specified sets positively invariant [7]. Prajna and Jadbabaie proposed a similar concept called barrier certificate [8], [9]. A control barrier function-based approach has been applied to differentiable nonlinear systems [10], [11], [12], such as adaptive cruise control [13]. The framework of a control system

The associate editor coordinating the review of this manuscript and approving it for publication was Wai-Keung Fung.

that simultaneously considers both control specification and safety is presented in [14]. It has also been applied to discrete time systems [15], [16], possibly with Gaussian noise [17].

To ensure the safety of mobile robots sharing a workspace, communication among them should be considered. For example, a distributed fault-tolerant controller was designed in [18], where each robot contained an observer that estimated the states of the other robots using data received from the other robots and a fault model. However, the behavior of the robots is not guaranteed in the case of communication faults. A method for asymptotically stabilizing a T-S fuzzy system connected to a delayed network was proposed in [19]. The tracking control of a system whose dynamics is unknown and time-varying was considered in [20]. However, the safety of the controlled systems has not been discussed in these studies. Consequently, safety against sporadic disturbances such as packet dropouts has not been ensured.

On the other hand, the design of finite-state controllers is crucial for mobile robots. This is because the verification and validation of controllers with infinite numbers of states often incur considerable cost. Subsequently, an abstraction-based formal synthesis was proposed [21]. It aims to provide correct-by-design controllers for infinite and nondeterministic transition systems by using finite-abstracted models. It starts from the analysis of finite transition systems, possibly with nondeterministic transitions, to verify the satisfaction of control specifications [22], [23]. To discuss the relationship between a controlled system and desired behaviors (i.e., control specifications), simulation relations and alternating simulation relations have been introduced [24]. These have been extended to approximate simulation relations and approximate alternating simulation relations for the finite abstraction of infinite transition systems, such as hybrid systems [25], [26], [27], possibly with bounded disturbances [28], and cyber-physical systems [29], [30]. Recently, the authors extended control barrier functions to symbolic control barrier functions for abstraction-based formal synthesis, in which set invariance is enforced on a (physical) plant with a finite abstracted controller, despite unpredictable packet dropouts [31]. The procedure for designing the controller in [31] is as follows:

1) Construct a finite abstracted model of each mobile robot;
2) Compose them to obtain the entire system;
3) Design an (original) abstraction-based controller for the entire system without considering the safety;
4) Eliminate unsafe transitions in the entire system computed by the symbolic control barrier function.

However, there is a problem with the composition of the abstracted models. If each finite abstracted model of a mobile robot has X states and there are N mobile robots in the workspace, the entire system has $O(X^N)$ states, which have a negative impact on the computation.

Instead of composing each mobile robot to consider the entire system, another approach based on *shields* was proposed to ensure safety [32], [33]. Figure 1 illustrates the role

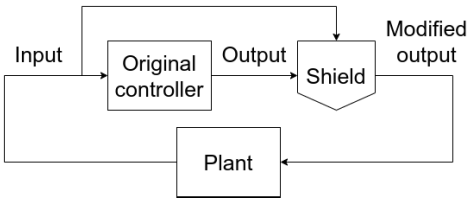

FIGURE 1. Summary of the shield attached to the system.

of the shield. The shield was placed between the original controller and the plant. It monitors the input and output of the original controller and overwrites the output (i.e., a control input) with a safe one to prevent violation of the safety specification. If the output is safe, the shield does not overwrite it. With the introduction of the shield, the composition of all robots is not required. This is because the shield is attached to each mobile robot, and the safety of each robot is ensured by it. Shields have been applied to a variety of complicated systems such as multi-agent systems [34], human-interactive robotics [35], [36], dynamic traffic light controllers [37], and security-aware path planners [38]. In existing literature, shields are designed based on the premise that all transitions in the entire system are deterministic. The premise is restrictive because it ensures that all communication packets among the robots are always delivered successfully. If there is an unpredictable packet dropout, safety is not guaranteed.

In this study, because we designed a shield based on an alternating simulation relation between a mobile robot and its abstracted model, safety is still assured despite unpredictable packet dropouts, as in the case of [31]. The shield proposed in this study is called the *alternating simulation-based shield (AS-Shield)*. The procedure for designing the proposed controller is as follows:

1) Construct a finite abstracted model of each mobile robot;
2) Design an original abstraction-based controller for each mobile robot without considering safety;
3) Design an AS-Shield for each mobile robot to avoid collisions and generate avoidance trajectories;
4) Attach the AS-Shield to the original abstraction-based controller for each mobile robot.

The AS-Shield is obtained from a finite transition system such that all transitions in the AS-Shield are safe. To eliminate unsafe transitions, a symbolic control barrier function was used in the design of the AS-Shield. We theoretically ensured that the shield-attached controlled system is safe. Thus, the mobile robots do not collide and arrive at their target states successfully. As well as existing literature on shields, we do not compose all mobile robots in the workspace; therefore, the number of states in the entire system is reduced from $O(X^N)$ to $O(NX)$. We confirmed the validity of the proposed method using a robot simulator.

In summary, contributions of this paper are as follows:

1) A shield, whose safety under nondeterministic transitions (sporadic disturbances) is guaranteed by alternating simulation relations, is designed;

2) Our proposed approach can reduce the computation time required to obtain the proposed controller compared to the controller designed in [31];
3) A framework of distributed (decentralized) control with the proposed shield is presented for a mobile robot example, and its validity is confirmed using a robot simulator.

The remainder of this paper is organized as follows. We review the fundamental notions of abstraction-based formal synthesis in Section II. In Section III, the AS-Shield is defined and the satisfaction of the control and safety specifications is theoretically assured. We design AS-Shields for mobile robots in Section IV, where the mobile robots are modeled in Section IV-A, an AS-Shield for collision avoidance is designed in Section IV-B, and one for collision and deadlock avoidance is described in Section IV-C. The proposed method is evaluated in Section V, and we conclude this paper in Section VI.

II. PRELIMINARIES

A. NOTATIONS AND DEFINITIONS

We use the notations $\mathbb{R}$, $\mathbb{Z}$, $\mathbb{R}_{>0}$, and $\mathbb{Z}_{>0}$ to describe sets of real numbers, integers, positive real numbers, and positive integers, respectively. For $x \in \mathbb{R}^n$ with $n \in \mathbb{Z}_{>0}$, $|x|$ indicates the ∞-norm. For any $a \in \mathbb{R}$ and any $b \in \mathbb{R}$ such that $a < b$, $(a, b] \subseteq \mathbb{R}$ is defined by $(a, b] = \{x \in \mathbb{R} \mid a < x \leq b\}$. For any $a \in \mathbb{Z}$ and any $b \in \mathbb{Z}$ such that $a < b$, $[a; b] \subseteq \mathbb{Z}$ is defined by $[a; b] := \{x \in \mathbb{Z} \mid a \leq x \leq b\}$. For a given set A, denoted by 2^A is the power set of A, and denoted by $|A|$ is the number of elements.

In the following, we review definitions related to abstraction-based formal synthesis.

Definition 1 (System [22]): System S is a tuple (X, X_0, U, r), where

- X is a set of states;
- $X_0 \subseteq X$ is a set of initial states;
- U is a set of inputs;
- $r : X \times U \to 2^X$ is a transition map.

For any $x \in X$, let $U(x) := \{u \in U \mid r(x, u) \neq \emptyset\}$.

Let $S_1 = (X_1, X_{10}, U_1, r_1)$ and $S_2 = (X_2, X_{20}, U_2, r_2)$ be two systems. For a relation $R \subseteq X_1 \times X_2 \times U_1 \times U_2$ over the state sets X_1, X_2 and the input sets U_1, U_2, denoted by $R_X \subseteq X_1 \times X_2$ is a projection of R to the state sets X_1, X_2 as follows:

$$R_X := \{(x_1, x_2) \in X_1 \times X_2 \mid \exists u_1 \in U_1, \exists u_2 \in U_2 : (x_1, x_2, u_1, u_2) \in R\}.$$

Definition 2 (Alternating simulation relation [22]): Let $S_1 = (X_1, X_{10}, U_1, r_1)$ and $S_2 = (X_2, X_{20}, U_2, r_2)$ be two systems. We call the relation $R \subseteq X_1 \times X_2 \times U_1 \times U_2$ an alternating simulation relation (ASR) from S_1 to S_2 if the following two conditions hold:

1) $\forall x_{10} \in X_{10}, \exists x_{20} \in X_{20} : (x_{10}, x_{20}) \in R_X$; and
2) $\forall x_1 \in X_1, \forall x_2 \in X_2, \forall u_1 \in U_1(x_1), \exists u_2 \in U_2(x_2)$:
$$(x_1, x_2, u_1, u_2) \in R \Rightarrow \forall x_2' \in r_2(x_2, u_2), \exists x_1' \in r_1(x_1, u_1) : (x_1', x_2') \in R_X.$$

Definition 3 (Controller [22]): Let $S_p = (X_p, X_{p0}, U_p, r_p)$ and $S_c = (X_c, X_{c0}, U_c, r_c)$ be two systems, and consider a relation $R_c \subseteq X_c \times X_p \times U_c \times U_p$. We call the pair (S_c, R_c) a controller for S_p if R_c is an ASR from S_c to S_p.

Definition 4 (System composition [22]): Let $S_1 = (X_1, X_{10}, U_1, r_1)$ and $S_2 = (X_2, X_{20}, U_2, r_2)$ be two systems, and let $R \subseteq X_1 \times X_2 \times U_1 \times U_2$ be a relation. We define the composition of S_1 and S_2 with respect to R, denoted by $S := S_1 \times_R S_2 = (X, X_0, U, r)$, where

- $X := X_1 \times X_2$;
- $X_0 := (X_{10} \times X_{20}) \cap R_X$;
- $U := U_1 \times U_2$;
- $r : X \times U \to 2^X$ is defined as follows: $(x_1', x_2') \in r((x_1, x_2), (u_1, u_2))$ iff
$$\left[x_1' \in r_1(x_1, u_1)\right] \wedge \left[x_2' \in r_2(x_2, u_2)\right] \wedge [(x_1, x_2, u_1, u_2) \in R] \wedge \left[(x_1', x_2') \in R_X\right].$$

If $R = X_1 \times X_2 \times U_1 \times U_2$, we simply denote the composed system by $S_1 \times S_2$.

B. ABSTRACTION-BASED FORMAL SYNTHESIS [21]

Let us consider a (physical) plant

$$S = (X, X_0, U, r), \tag{1}$$

where X is possibly infinite, and its finite-state abstracted plant model

$$\hat{S} = (\hat{X}, \hat{X}_0, \hat{U}, \hat{r}) \tag{2}$$

such that there exists an ASR

$$R \subseteq \hat{X} \times X \times \hat{U} \times U \tag{3}$$

from $\hat{S}$ to S. Assume that we have obtained a finite-state controller for $\hat{S}$ denoted by the pair $(\hat{S}_C, \hat{R}_C)$, where

$$\hat{S}_C = (\hat{X}_C, \hat{X}_{C0}, \hat{U}_C, \hat{r}_C) \tag{4}$$

is the system describing desired behaviors of $\hat{S}$, and the relation

$$\hat{R}_C \subseteq \hat{X}_C \times \hat{X} \times \hat{U}_C \times \hat{U} \tag{5}$$

is an ASR from $\hat{S}_C$ to $\hat{S}$. Then, we have the following theorem.

Theorem 1 (Abstraction-based controller [21]): The pair (S_C, R_C) is a controller for S, where

$$S_C := \hat{S}_C \times_{\hat{R}_C} \hat{S} = (X_C, X_{C0}, U_C, r_C) \tag{6}$$

and the relation $R_C \subseteq X_C \times X \times U_C \times U$ is given by

$$R_C := \{((\hat{x}_C, \hat{x}), x, (\hat{u}_C, \hat{u}), u) \in X_C \times X \times U_C \times U \mid [(\hat{x}, x, \hat{u}, u) \in R] \wedge [(\hat{x}_C, \hat{x}, \hat{u}_C, \hat{u}) \in \hat{R}_C]\}. \tag{7}$$

It is easily confirmed that R_C defined by (7) is an ASR from S_C to S. Theorem 1 implies that we have a (refined) controller for S by composing the finite transition systems $\hat{S}_C$ and $\hat{S}$.

C. SYMBOLIC CONTROL BARRIER FUNCTIONS

For a given safe set $\mathcal{X} \subseteq X$ of (1), we consider a mapping $B : X \to \mathbb{R}$ such that $\mathcal{X}$ is described as follows [15], [16]:

$$\mathcal{X} := \{x \in X \mid B(x) \geq 0\}. \tag{8}$$

Definition 5 (Safe controller [31]): Let $S_p = (X_p, X_{p0}, U_p, r_p)$ and $S_c = (X_c, X_{c0}, U_c, r_c)$ be two systems, $R_c \subseteq X_c \times X_p \times U_c \times U_p$ be an ASR from S_c to S_p, and $B_p : X_p \to \mathbb{R}$ be a mapping induced by a safe set $\mathcal{X}_p \subseteq X_p$. We call the pair (S_c, R_c) a *safe* controller with respect to $\mathcal{X}_p$ if the following condition is satisfied:

$$\begin{aligned}\forall (x_c, x_p, u_c, u_p) \in R_c \text{ s.t. } u_c \in U_c(x_c) :\\ x_p \in \mathcal{X}_p \Rightarrow r_p(x_p, u_p) \subseteq \mathcal{X}_p.\end{aligned} \tag{9}$$

To consider the safety of (1) with its finite abstracted model (2), we define the mapping $\hat{B} : \hat{X} \to \mathbb{R}$ induced by $B : X \to \mathbb{R}$ as follows:

$$\hat{B}(\hat{x}) = \inf_{x \in X \text{ s.t. } (\hat{x}, x) \in R_X} B(x). \tag{10}$$

For mapping $\hat{B} : \hat{X} \to \mathbb{R}$, the following set $\hat{\mathcal{X}}$ is called a safe set of the abstracted plant model $\hat{S}$:

$$\hat{\mathcal{X}} := \{\hat{x} \in \hat{X} \mid \hat{B}(\hat{x}) \geq 0\}. \tag{11}$$

Definition 6 (Symbolic control barrier function [31]): Mapping $\hat{B} : \hat{X} \to \mathbb{R}$ is said to be a symbolic control barrier function if the following condition is satisfied:

$$\forall \hat{x} \in \hat{\mathcal{X}},\ \exists \hat{u} \in \hat{U}(\hat{x}) :\ \Delta\hat{B}(\hat{x}, \hat{u}) + \hat{B}(\hat{x}) \geq 0, \tag{12}$$

where

$$\Delta\hat{B}(\hat{x}, \hat{u}) := \min_{\hat{x}' \in \hat{r}(\hat{x}, \hat{u})} \hat{B}(\hat{x}') - \hat{B}(\hat{x}). \tag{13}$$

For any $\hat{x} \in \hat{\mathcal{X}}$, let

$$\hat{\mathcal{U}}^{\hat{B}}(\hat{x}) := \{\hat{u} \in \hat{U}(\hat{x}) \mid \Delta\hat{B}(\hat{x}, \hat{u}) + \hat{B}(\hat{x}) \geq 0\}. \tag{14}$$

We have the following theorem [31].

Theorem 2 (Abstraction-based safe controller [31]): Let $B : X \to \mathbb{R}$ be the mapping describing the safe set (8) to be considered, and the mapping $\hat{B} : \hat{X} \to \mathbb{R}$ is computed using (10). If a safe controller for (2) exists with respect to (11), the refined controller (S_C, R_C) obtained by Theorem 1 is a safe controller with respect to (8) for (1).

Theorem 2 shows that the safety specification is enforced on (physical) plant (1) by considering the conservative mapping (10) and a symbolic control barrier function with respect to it. We verified the (physical) plant's satisfaction with the safety specifications using its finite-abstracted model.

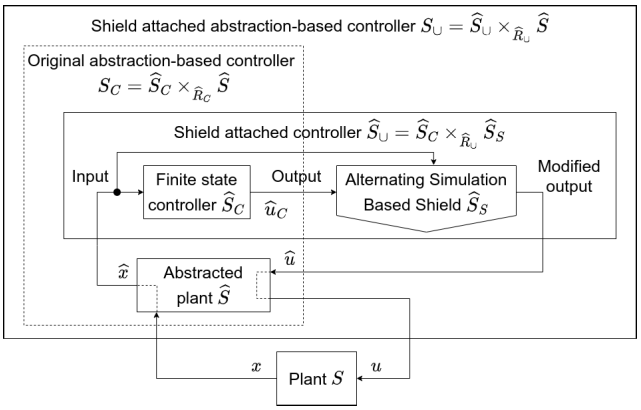

FIGURE 2. Summary of the alternating simulation-based shield attached to the (physical) plant.

III. ALTERNATING SIMULATION-BASED SHIELD

In this study, we introduce a novel shield called an alternating simulation-based shield (AS-Shield) to leverage abstraction-based formal synthesis. Figure 2 shows the configuration of the shield-attached abstraction-based controller. The AS-Shield $\hat{S}_S$ monitors the input and output of the finite-state controller $(\hat{S}_C, \hat{R}_C)$ and overwrites the output (i.e., the control input) with a safe one to prevent a violation of the safety specification if necessary. In the following, we present the details of the design of the shield-attached abstraction-based controller.

The AS-Shield is constructed such that all transitions in it are safe. Therefore, the AS-Shield was constructed using a symbolic control barrier function.

Definition 7 (Alternating simulation-based shield): Assume that the mapping $\hat{B} : \hat{X} \to \mathbb{R}$ computed by (10) is a symbolic control barrier function for (2). Then, the system

$$\hat{S}_S = (\hat{X}_S, \hat{X}_{S0}, \hat{U}_S, \hat{r}_S), \tag{15}$$

is called an alternating simulation-based shield (AS-Shield) with respect to $\hat{B}$, where $\hat{X}_S = \hat{X}$, $\hat{X}_{S0} = \hat{X}_0$, $\hat{U}_S = \hat{U}$, and $\hat{r}_S : \hat{X}_S \times \hat{U}_S \to 2^{\hat{X}_S}$ satisfies the following condition:

$$\begin{aligned}&\forall \hat{x}_S \in \hat{X}_S, \forall \hat{u}_S \in \hat{U}_S :\\ &\hat{u}_S \in \hat{U}_S(\hat{x}_S) \Rightarrow [\hat{u}_S \in \hat{\mathcal{U}}^{\hat{B}}(\hat{x}_S)] \wedge [\hat{r}_S(\hat{x}_S, \hat{u}_S) = \hat{r}(\hat{x}_S, \hat{u}_S)].\end{aligned} \tag{16}$$

Note that $\hat{S}_S$ is designed with $\hat{S}$ and is not dependent on $\hat{S}_C$.

To attach the AS-Shield to the abstraction-based controller, we introduce the notion of a subtransition system.

Definition 8 (Subtransition system): Let $S_1 = (X_1, X_{10}, U_1, r_1)$ and $S_2 = (X_2, X_{20}, U_2, r_2)$ be two systems. We say that S_1 is a subtransition system of S_2 if the following conditions hold:

- $X_1 \subseteq X_2$;
- $X_{10} \subseteq X_{20}$;
- $U_1 \subseteq U_2$;
- $\forall x \in X_1, \forall u \in U_1(x) : r_1(x, u) = r_2(x, u)$.

If S_1 is a subtransition system of S_2, we use the notation $S_1 \subseteq S_2$.
The $\hat{S}_S$ defined by (15) is a subtransition system of $\hat{S}$. For subtransition systems, we have the following lemma.

Lemma 1: Let $S_p = (X_p, X_{p0}, U_p, r_p)$ and $S_c = (X_c, X_{c0}, U_c, r_c)$ be two systems such that $S_c \subseteq S_p$. The following relation $R_c \subseteq X_c \times X_p \times U_c \times U_p$ is an ASR from S_c to S_p:

$$R_c = \{(x_c, x_p, u_c, u_p) \in X_c \times X_p \times U_c \times U_p \mid [x_c = x_p] \wedge [u_c = u_p]\}. \tag{17}$$

The proof of Lemma 1 is shown in Appendix A.

We attach the AS-Shield to the abstraction-based controller as follows.

Definition 9 (Shield-attached controller): Assume that $\hat{B} : \hat{X} \to \mathbb{R}$ computed by (10) is a symbolic control barrier function for (2) and that $\hat{S}_C$ defined by (4) satisfies $\hat{S}_C \subseteq \hat{S}$. Let $\hat{S}_S = (\hat{X}_S, \hat{X}_{S0}, \hat{U}_S, \hat{r}_S)$ be an AS-Shield with respect to $\hat{B}$. Then, the pair

$$(\hat{S}_\cup, \hat{R}_\cup) = ((\hat{X}_\cup, \hat{X}_{\cup 0}, \hat{U}_\cup, \hat{r}_\cup), \hat{R}_\cup) \tag{18}$$

is called a *shield-attached controller* induced by $\hat{S}_C$ and $\hat{S}_S$, where $\hat{X}_\cup = \hat{X}, \hat{X}_{\cup 0} = \hat{X}_0 \cap \hat{\mathcal{X}}, \hat{U}_\cup = \hat{U}, \hat{r}_\cup : \hat{X}_\cup \times \hat{U}_\cup \to 2^{\hat{X}_\cup}$ is defined as follows:

$$\hat{r}_\cup(\hat{x}_\cup, \hat{u}_\cup) = (\hat{r}_C(\hat{x}_\cup, \hat{u}_\cup) \cap \hat{r}_S^m(\hat{x}_\cup, \hat{u}_\cup)) \cup \hat{r}_S(\hat{x}_\cup, \hat{u}_\cup), \tag{19}$$

where $\hat{r}_S^m : \hat{X}_\cup \times \hat{U}_\cup \to 2^{\hat{X}_\cup}$ is defined by

$$\hat{r}_S^m(\hat{x}_S^m, \hat{u}_S^m) = \begin{cases} \hat{r}(\hat{x}_S^m, \hat{u}_S^m) & \text{if } \hat{u}_S^m \in \hat{\mathcal{U}}^{\hat{B}}(\hat{x}_S^m); \\ \emptyset & \text{otherwise,} \end{cases} \tag{20}$$

and $\hat{R}_\cup \subseteq \hat{X}_\cup \times \hat{X} \times \hat{U}_\cup \times \hat{U}$ is defined by

$$\hat{R}_\cup = \{(\hat{x}_\cup, \hat{x}, \hat{u}_\cup, \hat{u}) \in \hat{X}_\cup \times \hat{X} \times \hat{U}_\cup \times \hat{U} \mid [\hat{x}_\cup = \hat{x}] \wedge [\hat{u}_\cup = \hat{u}]\}. \tag{21}$$

Lemma 2: The pair $(\hat{S}_\cup, \hat{R}_\cup)$ defined by (18) is a safe controller for $\hat{S}$ with respect to $\hat{B}$.
The proof of Lemma 2 is shown in Appendix B.

Finally, we have the following main theorem.

Theorem 3 (Shield-attached abstraction-based controller): The refined controller $(S_\cup, R_\cup)$ is a safe controller with respect to (8) for (1), where

$$S_\cup := \hat{S}_\cup \times_{\hat{R}_\cup} \hat{S} = (X_\cup, X_{\cup 0}, U_\cup, r_\cup) \tag{22}$$

and $R_\cup \subseteq X_\cup \times X \times U_\cup \times U$ is defined as follows:

$$R_\cup = \{((\hat{x}_\cup, \hat{x}), x, (\hat{u}_\cup, \hat{u}), u) \in X_\cup \times X \times U_\cup \times U \mid [(\hat{x}, x, \hat{u}, u) \in R] \wedge [(\hat{x}_\cup, \hat{x}, \hat{u}_\cup, \hat{u}) \in \hat{R}_\cup]\}. \tag{23}$$

The proof of Theorem 3 is shown in Appendix C.

It is concluded that the shield-attached abstraction-based controller enforces the desired behaviors described by (4) and safety by (15) on (1).

In the next section, we concretely design AS-Shields for a mobile robot such that the robot moves to its destination without colliding with obstacles or other robots in a workspace.

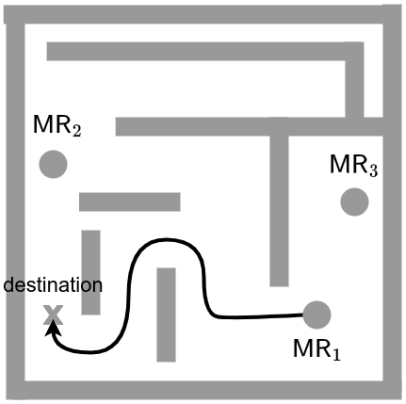

FIGURE 3. An example of multiple mobile robots ($N = 3$).

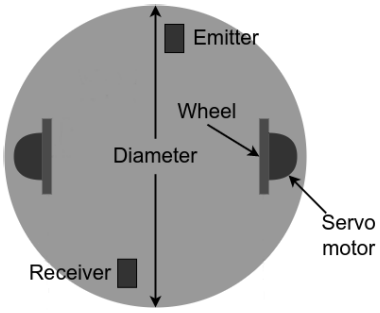

FIGURE 4. The configuration of each mobile robot.

IV. AS-SHIELDS FOR A MOBILE ROBOT

Let $N \in \mathbb{Z}_{>0}$ be the number of mobile robots in a workspace. We use the notation MR_i ($i \in [1; N]$) to specify the i-th mobile robot. Figure 3 shows an example of the entire system with $N = 3$. The objective is to move each robot to its destination without colliding with obstacles (walls) or other robots. The configuration of each robot was the same as that shown in Fig. 4. Each wheel on the robot is controlled using a servo motor. In addition, each robot is equipped with an emitter and receiver for broadcast communication.

Each robot is modeled as a discrete-time system, and all robots are time-synchronized. In each time step, the following computations were performed:

1) Receive information from the other robots;
2) Determine a control input by an original abstraction-based controller or an AS-Shield;
3) Drive the servo motors to rotate the wheels;
4) Transit to the next state on the original abstraction-based controller and AS-Shield;
5) Send information to the other robots.

Computations 1)-5) are executed only once in each time step. Even when a packet dropout occurs, the information is not sent again in a single time step. Instead, we send (possibly updated) information at the next time step.

In the following, we impose the following assumptions:

- The workspace where the robots move is bounded and the positions of the obstacles are known;

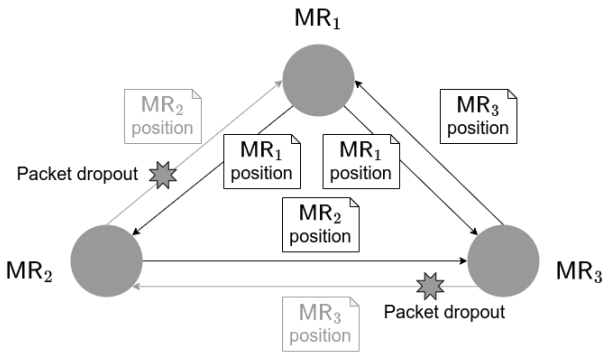

FIGURE 5. An example of packet dropouts ($N = 3$).

- The number of robots in the workspace is known;
- The communication among robots sometimes fails randomly due to packet dropouts;
- Each robot sends (at least) its current and next position at each time step.

Recall that communication among robots is broadcast; that is, each robot sends information to all other robots. When a packet dropout occurs, the information in the dropped packet is not delivered. For example, in Fig. 5, the position of MR_2 is not sent to MR_1 and that of MR_3 is not sent to MR_2.

When a packet dropout occurs, the robot does not reach the *next* position at the next time step because it does not leave the current position to prevent collisions caused by incomplete information (refer to (24)). Then, the *next* position sent at each time step is the position of the mobile robot at the next time step, assuming that the robot successfully receives all packets from other robots. Thus, the *next* position sent by each robot does not always imply that the robot is in the *next* position at the next time step. Nevertheless, no collisions are assured because AS-Shields prohibit the proceeding to the *current* (and next) positions of the other robots. Details of the design of the AS-Shields are presented in Sections IV-B and IV-C. Each robot detects packet dropouts by comparing the number of received packets with the number of other robots in the workspace.

We designed a homogeneous controller for MR_1, MR_2, …, MR_N. Thus, we designed a controller for a single mobile robot MR_i ($i \in [1; N]$) in Sections IV-A-IV-C.

A. MODELING AND ABSTRACTION

First, we modeled the dynamics of MR_i. Because it is connected to a network to communicate with other robots, we consider the two-wheel robot (denoted by S^r) and the network (denoted by S^n), and compose them to obtain a model of MR_i. The system $S^r = (X^r, X_0^r, U^r, r^r)$ is given by $X^r = \{(x^r, y^r, \theta^r) \mid x^r \in \mathbb{R},\ y^r \in \mathbb{R},\ \theta^r \in (-\pi, \pi]\}$, $X_0^r = X^r$, and $U^r = \{(a, b) \mid a \in \mathbb{R},\ b \in (-\pi, \pi]\}$. Here, the state $(x^r, y^r, \theta^r) \in X^r$ consists of x (horizontal) and y (vertical) coordinates of the center of the robot, and the angle of the robot from x-axis. In addition, input $(a, b) \in U^r$ consists of the translation distance and rotation angle. For simplicity, it was assumed that the robot could translate or rotate exclusively. In other words, it cannot simultaneously perform translational and rotational motions. Subsequently, the transition map $r^r : X^r \times U^r \to X^r$ is given by

$$
r^r((x^r, y^r, \theta^r), u^r) = \begin{cases} \{(x^r + a\cos\theta^r,\ y^r + a\sin\theta^r,\ \theta^r)\} \\ \qquad \text{if } u^r = (a, 0); \\ \{(x^r,\ y^r,\ \theta^r + b)\} \\ \qquad \text{if } u^r = (0, b) \text{ and } \theta^r + b \in (-\pi, \pi]; \\ \{(x^r,\ y^r,\ \theta^r + b - 2\pi)\} \\ \qquad \text{if } u^r = (0, b) \text{ and } \theta^r + b > \pi; \\ \{(x^r,\ y^r,\ \theta^r + b + 2\pi)\} \\ \qquad \text{if } u^r = (0, b) \text{ and } \theta^r + b \le -\pi; \\ \emptyset \qquad \text{otherwise.} \end{cases}
$$

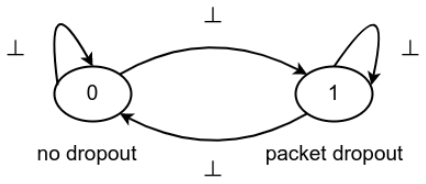

FIGURE 6. Automaton model of dynamics of the communication channel.

The dynamics of the communication network were modeled using an automaton, as shown in Fig. 6. The system $S^n = (X^n, X_0^n, U^n, r^n)$ is given by $X^n = \{0, 1\}$, $X_0^n = \{0\}$, $U^n = \{\bot\}$, and $r^n : X^n \times U^n \to 2^{X^n}$ is given by the automaton. To address packet dropouts, a "safety mechanism" was embedded in each robot MR_i. In other words, if a packet dropout is detected (i.e., $x^n = 1$), the mobile robot MR_i does not accept any input for translation. This is because MR_i may collide owing to incomplete information on the positions of the other robots. By contrast, any input for rotation is always accepted. Consequently, the plant model of MR_i is given by a nondeterministic transition system $S = (X, X_0, U, r)$, where $X = X_0 = X^r$, $U = U^r$, and the transition map $r : X \times U \to 2^X$ given by

$$
r((x, y, \theta), u) = \begin{cases} r^r((x, y, \theta), u) \cup \{(x, y, \theta)\} & \text{if } u \text{ is for translation;} \\ r^r((x, y, \theta), u) & \text{if } u \text{ is for rotation.} \end{cases} \tag{24}
$$

Second, we considered an abstracted model of MR_i. The workspace was separated into a finite number of grids, as shown in Fig. 7. In each time step, we move MR_i to the next grid, rotate it by $\pm\pi/2$ [rad], or maintain its current state. Let the grid size be $g \in \mathbb{R}_{>0}$ [m] and let the number of grids on the x-axis (resp. y-axis) be $2m + 1$ (resp. $2n + 1$) with $m \in \mathbb{Z}_{>0}$ (resp. $n \in \mathbb{Z}_{>0}$). Then, the abstracted model $\hat{S} = (\hat{X}, \hat{X}_0, \hat{U}, \hat{r})$ is given by $\hat{X} = \{(g\hat{x}, g\hat{y}, \hat{\theta}) \mid \hat{x} \in [-m; m],\ \hat{y} \in [-n; n],\ \hat{\theta} \in \{-\pi/2, 0, \pi/2, \pi\}\}$, $\hat{X}_0 = \hat{X}$, and $\hat{U} = \{\hat{u}_t^+, \hat{u}_t^-, \hat{u}_\epsilon, \hat{u}_r^+, \hat{u}_r^-\}$. Each input $\hat{u}_t^+$ (resp. $\hat{u}_t^-$) implies $+g$

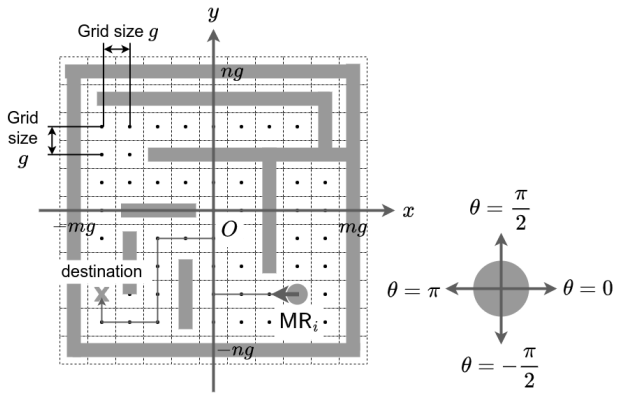

FIGURE 7. The workspace is separated into grids.

[m] (resp. $-g$ [m]) translation and $\hat{u}_r^+$ (resp. $\hat{u}_r^-$) implies $+\pi/2$ [rad] (resp. $-\pi/2$ [rad]) rotation, and $\hat{u}_\epsilon$ implies no motion. The input $\hat{u} \in \hat{U}$ is converted to $u \in U$ with the following map $I : \hat{U} \to U$:

$$I(\hat{u}) = \begin{cases} (g, 0) & \text{if } \hat{u} = \hat{u}_t^+; \\ (-g, 0) & \text{if } \hat{u} = \hat{u}_t^-; \\ (0, \pi/2) & \text{if } \hat{u} = \hat{u}_r^+; \\ (0, -\pi/2) & \text{if } \hat{u} = \hat{u}_r^-; \\ (0, 0) & \text{if } \hat{u} = \hat{u}_\epsilon. \end{cases} \tag{25}$$

The transition map $\hat{r} : \hat{X} \times \hat{U} \to 2^{\hat{X}}$ is defined as follows:

$$\hat{r}((g\hat{x}, g\hat{y}, \hat{\theta}), \hat{u}) = \begin{cases} \{(g(\hat{x}+1), g\hat{y}, \hat{\theta}), (g\hat{x}, g\hat{y}, \hat{\theta})\} \\ \quad \text{if } [\hat{u} = \hat{u}_t^+ \wedge \hat{\theta} = 0] \vee [\hat{u} = \hat{u}_t^- \wedge \hat{\theta} = \pi]; \\ \{(g(\hat{x}-1), g\hat{y}, \hat{\theta}), (g\hat{x}, g\hat{y}, \hat{\theta})\} \\ \quad \text{if } [\hat{u} = \hat{u}_t^- \wedge \hat{\theta} = 0] \vee [\hat{u} = \hat{u}_t^+ \wedge \hat{\theta} = \pi]; \\ \{(g\hat{x}, g(\hat{y}+1), \hat{\theta}), (g\hat{x}, g\hat{y}, \hat{\theta})\} \\ \quad \text{if } [\hat{u} = \hat{u}_t^+ \wedge \hat{\theta} = \frac{\pi}{2}] \vee [\hat{u} = \hat{u}_t^- \wedge \hat{\theta} = -\frac{\pi}{2}]; \\ \{(g\hat{x}, g(\hat{y}-1), \hat{\theta}), (g\hat{x}, g\hat{y}, \hat{\theta})\} \\ \quad \text{if } [\hat{u} = \hat{u}_t^- \wedge \hat{\theta} = \frac{\pi}{2}] \vee [\hat{u} = \hat{u}_t^+ \wedge \hat{\theta} = -\frac{\pi}{2}]; \\ \{(g\hat{x}, g\hat{y}, \hat{\theta} + \pi/2)\} \\ \quad \text{if } \hat{u} = \hat{u}_r^+ \wedge \hat{\theta} \in \{-\frac{\pi}{2}, 0, \frac{\pi}{2}\}; \\ \{(g\hat{x}, g\hat{y}, -\pi/2)\} \\ \quad \text{if } \hat{u} = \hat{u}_r^+ \wedge \hat{\theta} = \pi; \\ \{(g\hat{x}, g\hat{y}, \hat{\theta} - \pi/2)\} \\ \quad \text{if } \hat{u} = \hat{u}_r^- \wedge \hat{\theta} \in \{0, \frac{\pi}{2}, \pi\}; \\ \{(g\hat{x}, g\hat{y}, \pi)\} \\ \quad \text{if } \hat{u} = \hat{u}_r^- \wedge \hat{\theta} = -\frac{\pi}{2}; \\ \{(g\hat{x}, g\hat{y}, \hat{\theta})\} \\ \quad \text{if } \hat{u} = \hat{u}_\epsilon. \end{cases} \tag{26}$$

Despite (26), $\hat{r}((g\hat{x}, g\hat{y}, \hat{\theta}), \hat{u}) = \emptyset$ if the state after the transition is out of the boundaries of the workspace or if MR_i tries to pass a position occupied by an obstacle during the transition. Ultimately, the state after the transition is computed using Algorithm 1.

Algorithm 1 Compute the state after the transition with consideration of the boundaries and the obstacle positions

Input: the current state $(g\hat{x}, g\hat{y}, \hat{\theta})$, the input $\hat{u}$, the set of obstacle positions $\hat{\mathcal{O}} \subseteq \{(g\hat{x}^o, g\hat{y}^o) \mid \hat{x}^o \in [-m; m],\ \hat{y}^o \in [-n; n]\}$

Output: the state after the transition

```
(gx̂′, gŷ′, θ̂′) ← r̂((gx̂, gŷ, θ̂), û) computed by (26)
if x̂′ < −m, m < x̂′, ŷ′ < −n, or n < ŷ′ then
    # out of the boundary
    return NULL
else if (gx̂′, gŷ′) ∈ Ô then
    # occupied by an obstacle
    return NULL
else
    return (gx̂′, gŷ′, θ̂′)
end if
```

Then, the following relation $R \subseteq \hat{X} \times X \times \hat{U} \times U$ is an ASR from $\hat{S}$ to S:

$$\begin{aligned} R = \{((g\hat{x}, g\hat{y}, \hat{\theta}), (x, y, \theta), \hat{u}, u) \in \hat{X} \times X \times \hat{U} \times U \mid \\ [|(g\hat{x}, g\hat{y}) - (x, y)| \le g/2] \wedge [|\hat{\theta} - \theta| \le 0.01] \wedge \\ [I(\hat{u}) = u]\}. \end{aligned} \tag{27}$$

We move MR_i to its target state. Let the target state be $(g\hat{x}_t, g\hat{y}_t, \hat{\theta}_t) \in \hat{X}$. To obtain a path to $(g\hat{x}_t, g\hat{y}_t, \hat{\theta}_t)$, we apply the Dijkstra algorithm and obtain the system $\hat{S}_C = (\hat{X}_C, \hat{X}_{C0}, \hat{U}_C, \hat{r}_C)$ where $\hat{X}_C = \hat{X}$, $\hat{X}_{C0} = \hat{X}_0$, $\hat{U}_C = \hat{U}$, and $\hat{r}_C : \hat{X}_C \times \hat{U}_C \to 2^{\hat{X}_C}$ is given by

$$\hat{r}_C(\hat{x}_C, \hat{u}_C) = \begin{cases} \hat{r}(\hat{x}_C, \hat{u}_C) & \text{if } \hat{u}_C = D(\hat{x}_C); \\ \emptyset & \text{otherwise,} \end{cases} \tag{28}$$

where $D : \hat{X}_C \to \hat{U}_C$ is the map indicating the first element of the input sequence obtained as a Dijkstra path[1] from $\hat{x}_C \in \hat{X}_C$ to $(g\hat{x}_t, g\hat{y}_t, \hat{\theta}_t) \in \hat{X}_C$.

Since $\hat{S}_C \subseteq \hat{S}$ holds, Lemma 1 shows that the following relation $\hat{R}_C \subseteq \hat{X}_C \times \hat{X} \times \hat{U}_C \times \hat{U}$ is an ASR from $\hat{S}_C$ to S:

$$\begin{aligned} \hat{R}_C = \{(\hat{x}_C, \hat{x}, \hat{u}_C, \hat{u}) \in \hat{X}_C \times \hat{X} \times \hat{U}_C \times \hat{U} \mid \\ [\hat{x}_C = \hat{x}] \wedge [\hat{u}_C = \hat{u}]\}. \end{aligned} \tag{29}$$

B. COLLISION AVOIDANCE

First, we defined a safe set $\mathcal{X} \subseteq X$. Let the radius of the robot be $r_d \in \mathbb{R}_{>0}$. To prevent collisions, we have to maintain

[1] In this example, we use the Dijkstra algorithm for path planning. Other technologies, such as roadmap technologies and crisp logic, are also applicable to the design of $\hat{S}_C$.

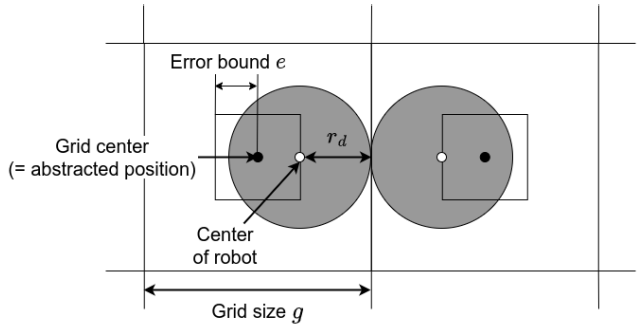

FIGURE 8. The worst case in terms of the distance between two robots.

distances from the other robots greater than $2r_d$. Therefore, the mapping $B : X \to \mathbb{R}$ is defined as follows:

$$B((x, y, \theta)) = \min_{(x_i, y_i) \in \mathcal{P} \cup \mathcal{P}'} |(x_i, y_i) - (x, y)| - 2r_d, \quad (30)$$

where $\mathcal{P} \subseteq \mathbb{R}^2$ (resp. $\mathcal{P}' \in \mathbb{R}^2$) is a set of current (resp. next) positions of the other robots MR_j ($j \neq i$).

Second, we computed $\hat{B}^\alpha : \hat{X} \to \mathbb{R}$ using (10). Recall that $\hat{S}$ is the abstracted model of S and that the workspace is divided into grids. Because all the positions in the same grid are abstracted to the central position of the grid, multiple mobile robots must not be located in the same grid. In addition, owing to sensor and actuator noises, errors exist between the actual position of the robot and the center of the grid. We assume that the position error is bounded by $e \in \mathbb{R}_{>0}$. Referring to Fig. 8, we set the grid size $g \in \mathbb{R}_{>0}$ such that the following condition is satisfied:

$$g > 2(r_d + e). \quad (31)$$

Then, it is guaranteed that each robot does not collide with the other robots as long as no grid contains multiple robots. Here, $\hat{B}^\alpha : \hat{X} \to \mathbb{R}$ is given by

$$\hat{B}^\alpha((g\hat{x}, g\hat{y}, \hat{\theta})) = \min_{(g\hat{x}_i, g\hat{y}_i) \in \hat{\mathcal{P}} \cup \hat{\mathcal{P}}'} \left|(g\hat{x}, g\hat{y}) - (g\hat{x}_i, g\hat{y}_i)\right| - 2(r_d + e), \quad (32)$$

where $\hat{\mathcal{P}} \subseteq \{(g\hat{x}, g\hat{y}) \mid \hat{x} \in [-m; m], \hat{y} \in [-n; n]\}$ (resp. $\hat{\mathcal{P}}' \subseteq \{(g\hat{x}, g\hat{y}) \mid \hat{x} \in [-m; m], \hat{y} \in [-n; n]\}$) is the set of the abstracted current (resp. next) positions of the other robots MR_j ($j \neq i$). Note that (10) is satisfied. The safe set $\hat{\mathcal{X}}^\alpha$ is given by

$$\hat{\mathcal{X}}^\alpha = \{(g\hat{x}, g\hat{y}, \hat{\theta}) \in \hat{X} \mid (g\hat{x}, g\hat{y}) \notin \hat{\mathcal{P}} \cup \hat{\mathcal{P}}'\}. \quad (33)$$

Third, we design an AS-Shield as follows:

$$\hat{S}_S^\alpha = (\hat{X}_S^\alpha, \hat{X}_{S0}^\alpha, \hat{U}_S^\alpha, \hat{r}_S^\alpha), \quad (34)$$

where $\hat{X}_S^\alpha = \hat{X}$, $\hat{X}_{S0}^\alpha = \hat{X}_0 \cap \hat{\mathcal{X}}^\alpha$, $\hat{U}_S^\alpha = \{\hat{u}_\epsilon\}$, and $\hat{r}_S^\alpha : \hat{X}_S^\alpha \times \hat{U}_S^\alpha \to 2^{\hat{X}_S^\alpha}$ is defined by

$$\hat{r}_S^\alpha(\hat{x}, \hat{u}_\epsilon) = \hat{r}(\hat{x}, \hat{u}_\epsilon) = \{\hat{x}\}. \quad (35)$$

Note that (16) is satisfied because input $\hat{u}_\epsilon$ is always safe.

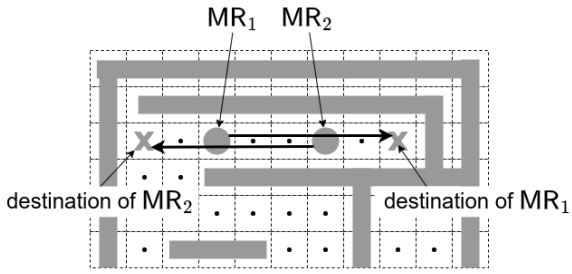

FIGURE 9. A case where a deadlock occurs if we block positions on original trajectories.

Finally, we obtain the shield-attached controller $(\hat{S}_\cup^\alpha, \hat{R}_\cup^\alpha)$ with $\hat{S}_S^\alpha$ and $\hat{B}^\alpha$ as in Definition 9, and the refined controller $(S_\cup^\alpha, R_\cup^\alpha)$ as in Theorem 3.

In fact, $\hat{S}_S^\alpha$ is not sufficient because of deadlocks. A case in which a deadlock occurs is presented in Section V-B.

C. COLLISION AND DEADLOCK AVOIDANCE

We consider not only collisions, but also deadlock avoidance. For collision avoidance, we used the discussion in Section IV-B. In addition, avoidance trajectories were generated for deadlock avoidance. Recall that all transitions in the AS-Shield must be safe. It is necessary to ensure that the robot is always safe while moving on an avoidance trajectory generated by the AS-Shield. Each robot must not proceed to the positions on the avoidance trajectories of the other robots. Additionally, each avoidance trajectory must not contain positions that are currently occupied by other robots. Based on the above discussion, we introduce the notion of blocked positions to ensure the safety of the avoidance trajectories.

Definition 10 (Blocked position): It is said that robot MR_i blocks a position $(g\hat{x}, g\hat{y})$ for $\hat{x} \in [-m; m]$ and $\hat{y} \in [-n; n]$ if MR_i prohibits any robot except MR_i entering the position $(g\hat{x}, g\hat{y})$. If a robot MR_i blocks a position $(g\hat{x}, g\hat{y})$, we say that $(g\hat{x}, g\hat{y})$ is a blocked position by MR_i or that $(g\hat{x}, g\hat{y})$ is blocked by MR_i.

It is assumed that each robot MR_i can block any position except for the positions currently occupied or already blocked by other robots.

Each robot uses the following blocked positions. When the AS-Shield in MR_i generates an avoidance trajectory, MR_i blocks all positions on the trajectory. Note that the other robots MR_j ($j \neq i$) do not enter these positions (prohibition to enter these positions is performed by the AS-Shield on MR_j). MR_i releases a blocked position when it passes that position.

We briefly explain why blocked positions are used not for an original trajectory to the destination $(g\hat{x}_t, g\hat{y}_t, \hat{\theta}_t)$ but for avoidance trajectories. Consider a simple case with two mobile robots in a workspace, as shown in Fig. 9. The initial position of MR_1 is at the entrance of a narrow passage, and the destination is at the dead end. The initial position of MR_2 is in the narrow passage, and the destination is out of it. Because MR_2 is on the MR_1's trajectory, MR_1 cannot block the positions. Similarly, MR_2 cannot block the positions, which implies a deadlock. Therefore, blocked positions are

used not for the original trajectories but for the avoidance trajectories.

Let $\hat{\mathcal{B}}^i$ be the set of positions on an avoidance trajectory generated by MR_i and $\hat{\mathcal{B}}^j$ ($j \neq i$) be the set of blocked positions by MR_j.[2] Let $\hat{\mathcal{B}} := \bigcup_{j\neq i} \hat{\mathcal{B}}^j$. If a position in $\hat{\mathcal{B}}^i$ exists that is blocked by another robot MR_j ($j \neq i$), MR_i may collide with MR_j while moving on the avoidance trajectory; that is, the AS-Shield is not obtained. Thus, the generated avoidance trajectory is valid only if all the positions on it are not blocked by any other robot. If a position on the avoidance trajectory is blocked by another robot, MR_i discards the avoidance trajectory and holds the position. Recall that computations 1)-5) are executed only once in a single time step. When a trajectory was discarded, another avoidance trajectory was generated at the next time step.

Ultimately, we obtain a mapping $\hat{B}^\beta : \hat{X} \to \mathbb{R}$ as follows:

$$\hat{B}^\beta((g\hat{x}, g\hat{y}, \hat{\theta})) = \min_{(g\hat{x}_i, g\hat{y}_i)\in\hat{\mathcal{P}}\cup\hat{\mathcal{P}}'\cup\hat{\mathcal{B}}} \left|(g\hat{x}, g\hat{y}) - (g\hat{x}_i, g\hat{y}_i)\right| - 2(r_d + e). \quad (36)$$

The safe set $\hat{\mathcal{X}}^\beta$ is given by

$$\hat{\mathcal{X}}^\beta = \{(g\hat{x}, g\hat{y}, \hat{\theta}) \in \hat{X} \mid (g\hat{x}, g\hat{y}) \notin \mathcal{P} \cup \mathcal{P}' \cup \hat{\mathcal{B}}\}. \quad (37)$$

Then, we design an AS-Shield

$$\hat{S}_S^\beta = (\hat{X}_S^\beta, \hat{X}_{S0}^\beta, \hat{U}_S^\beta, \hat{r}_S^\beta), \quad (38)$$

where $\hat{X}_S^\beta = \hat{X}$, $\hat{X}_{S0}^\beta = \hat{X}_0 \cap \hat{\mathcal{X}}^\beta$, $\hat{U}_S^\beta = \hat{U}$, and $\hat{r}_S^\beta : \hat{X}_S^\beta \times \hat{U}_S^\beta \to 2^{\hat{X}_S^\beta}$ is defined by

$$\hat{r}_S^\beta(\hat{x}, \hat{u}) = \begin{cases} \hat{r}(\hat{x}, \hat{u}) & \text{if } [\Delta\hat{B}^\beta(\hat{x}, \hat{u}) + \hat{B}^\beta(\hat{x}) \geq 0] \wedge [\hat{\mathcal{B}}^i \cap \hat{\mathcal{B}} = \emptyset]; \\ \{\hat{x}\} & \text{if } [\hat{\mathcal{B}}^i \cap \hat{\mathcal{B}} \neq \emptyset] \wedge [\hat{u} = \hat{u}_\epsilon]; \\ \emptyset & \text{otherwise.} \end{cases} \quad (39)$$

We obtain the shield-attached controller $(\hat{S}_\cup^\beta, \hat{R}_\cup^\beta)$ with $\hat{S}_S^\beta$ and $\hat{B}^\beta$ as in Definition 9, and the refined controller $(S_\cup^\beta, R_\cup^\beta)$ as in Theorem 3. In addition to collision avoidance, as in (34), AS-Shield (38) generates safe avoidance trajectories using the blocked positions.

We present an algorithm for generating an avoidance trajectory. In summary, we search for and set a temporal destination (i.e., the terminal of an avoidance trajectory) and generate a path to it. The temporal destination must be reachable from the current state without passing the obstacle positions or positions currently occupied or blocked by other robots. To consider the blocked positions, we modify the computation of the transition, as shown in Algorithm 2.

[2]Together with its current and next position, each robot MR_j sends positions included in $\hat{\mathcal{B}}^j$ to the other robots at each time step.

Algorithm 2 Compute the state after the transition by considering the currently occupied or blocked positions

Input: the current state $(g\hat{x}, g\hat{y}, \hat{\theta})$, the input $\hat{u}$, the set of obstacle positions $\hat{\mathcal{O}} \subseteq \{(g\hat{x}^o, g\hat{y}^o) \mid \hat{x}^o \in [-m; m],\ \hat{y}^o \in [-n; n]\}$, the set of current positions of the other robots $\hat{\mathcal{P}} \subseteq \{(g\hat{x}^p, g\hat{y}^p) \mid \hat{x}^p \in [-m; m],\ \hat{y}^p \in [-n; n]\}$, the set of next positions of the other robots $\hat{\mathcal{P}}' \subseteq \{(g\hat{x}^{p'}, g\hat{y}^{p'}) \mid \hat{x}^{p'} \in [-m; m],\ \hat{y}^{p'} \in [-n; n]\}$, the set of blocked positions by the other robots $\hat{\mathcal{B}} \subseteq \{(g\hat{x}^b, g\hat{y}^b) \mid \hat{x}^b \in [-m; m],\ \hat{y}^b \in [-n; n]\}$

Output: the state after the transition

$(g\hat{x}', g\hat{y}', \hat{\theta}') \leftarrow \hat{r}((g\hat{x}, g\hat{y}, \hat{\theta}), \hat{u})$ computed by (26)
if $\hat{x}' < -m$, $m < \hat{x}'$, $\hat{y}' < -n$, or $n < \hat{y}'$ **then**
  # out of the boundary
  return NULL
else if $(g\hat{x}', g\hat{y}') \in \hat{\mathcal{O}} \cup \hat{\mathcal{P}} \cup \hat{\mathcal{P}}' \cup \hat{\mathcal{B}}$ **then**
  # occupied or blocked by an obstacle or another robot
  return NULL
else
  return $(g\hat{x}', g\hat{y}', \hat{\theta}')$
end if

In addition, the temporal destination should not be on the trajectories of the other robots. Let $\hat{\mathcal{T}}^j$ ($j \neq i$) be the set of positions on the MR_j's trajectory and let $\hat{\mathcal{T}} := \bigcup_{j\neq i} \hat{\mathcal{T}}^j$. An algorithm to obtain an avoidance trajectory is presented in Algorithm 3.

In terms of the deviations from the original path to destination $(g\hat{x}_t, g\hat{y}_t, \hat{\theta}_t)$, avoidance trajectories with short path lengths are preferable. However, a livelock may exist if the shortest avoidance trajectory is always computed. Therefore, we introduce randomness by `rand()%10=0` in Algorithm 3; that is, if a shorter trajectory is found, it is accepted with a probability of 10%. We also introduce another randomness in triggering Algorithm 3 for the design of (38); that is, some robots sometimes do not generate their avoidance trajectories, even though a possible deadlock is detected. This is because they expected that the avoidance trajectories generated by other robots would resolve the deadlock.

Because the avoidance trajectories are safely generated, there is no deadlock, and all the robots successfully arrive at their destinations. The results are presented in Section V-C.

V. EVALUATION

In this section, we present the results of our proposed method. We set the size of the workspace as 2 [m] × 2 [m] and abstract it with 0.2 [m] × 0.2 [m] grids (i.e., $g = 0.2$ and $m = n = 5$). The Khepera IV was used as the mobile robot. Khepera IV was released by K-team [39], and its radius r_d is 70.40 [mm]. Khepera IV is equipped with two servo motors whose position error e is less than 0.02 [m], so (31) is satisfied.

To simulate Khepera IV robots, we used Webots software released by Cyberbotics [40] and randomly injected packet dropouts in the simulated communication. The simulation

Algorithm 3 Compute the avoidance trajectory

Input: the current state $(g\hat{x}, g\hat{y}, \hat{\theta})$, the input $\hat{u}$, the set of obstacle positions $\hat{\mathcal{O}} \subseteq \{(g\hat{x}^o, g\hat{y}^o) \mid \hat{x}^o \in [-m; m],\ \hat{y}^o \in [-n; n]\}$, the set of current positions of the other robots $\hat{\mathcal{P}} \subseteq \{(g\hat{x}^p, g\hat{y}^p) \mid \hat{x}^p \in [-m; m],\ \hat{y}^p \in [-n; n]\}$, the set of next positions of the other robots $\hat{\mathcal{P}}' \subseteq \{(g\hat{x}^{p'}, g\hat{y}^{p'}) \mid \hat{x}^{p'} \in [-m; m],\ \hat{y}^{p'} \in [-n; n]\}$, the set of blocked positions by the other robots $\hat{\mathcal{B}} \subseteq \{(g\hat{x}^b, g\hat{y}^b) \mid \hat{x}^b \in [-m; m],\ \hat{y}^b \in [-n; n]\}$, the set of positions on the other robots' trajectory $\hat{\mathcal{T}} \subseteq \{(g\hat{x}^t, g\hat{y}^t) \mid \hat{x}^t \in [-m; m],\ \hat{y}^t \in [-n; n]\}$

Output: the sequence of states indicating the avoidance trajectory

```
result.clear()  # output container
path_len ← INFINITY  # path length
(gx̂_d, gŷ_d, θ̂_d) ← (−1, −1, −1)  # temporal dest
Generate the abstracted plant model with Algorithm 2
Apply the Dijkstra algorithm on the generated model
for (gx̂^t, gŷ^t, θ̂^t) ∈ X̂ do
    flag ← TRUE
    if (gx̂^t, gŷ^t) ∈ Ô ∪ P̂ ∪ P̂' ∪ B̂ ∪ T̂ then
        # infeasible temporal destination
        flag ← FALSE
    end if
    if flag = TRUE then
        tmp_result ← the Dijkstra path obtained at step 5 from (gx̂, gŷ, θ̂) to (gx̂^t, gŷ^t, θ̂^t)
        if tmp_result is successfully obtained then
            if tmp_result.len() < path_len then
                # shorter avoidance trajectory found
                if rand()%10 = 0 then
                    # update the result
                    path_len ← tmp_result.len()
                    result ← tmp_result
                end if
            end if
        end if
    end if
end for
return result
```

TABLE 1. Simulation environment.

Software	Cyberbotics Webots R2022a, Date: December 17, 2021
CPU	AMD Ryzen 5 3600 6-Core Processor
GPU	NVIDIA GeForce GTX 1660 SUPER
Memory	Samsung M378A2G43AB3-CWE DDR4-3200 16GB × 2
OS	Ubuntu 20.04.4 LTS (Focal Fossa)
Graphics	NVIDIA Driver Version: 470.141.03, CUDA Version: 11.4

environment is shown in Table 1, and the initial and target positions are set as shown in Table 2.

A. CONVENTIONAL VS PROPOSED METHOD

Let us consider the conventional method, where all the robots in the workspace are composed, the abstracted model of the entire system is constructed, and unsafe transitions in it are eliminated. Let $N = 1$. Because a robot is located in a grid that is not occupied by an obstacle, the number of states in the entire system is the same as the number of grids not occupied by the obstacles, that is, 51 states in the case of Fig. 7. Each state has five transitions because $|\hat{U}| = 5$. Thus, 255 transitions were considered in the entire system. Let $N = 2$. Each robot was located in a grid that was not occupied by an obstacle or another robot. Thus, the number of states of the entire system was $51 \times 50 = 2550$. Each 2550 state had 25 transitions because $|\hat{U} \times \hat{U}| = 25$. Thus, there were 63750 transitions. In general, if there are N robots in the workspace, the number of states in the entire system is calculated as $51!/(51-N)!$ and that of the transitions is by $5^N \times 51!/(51-N)!$. The number of states and transitions in the entire system increased exponentially, as shown in Fig. 10.

TABLE 2. Simulation parameters ($N = 3$).

Robot	Initial position	Target position
MR_1	$(0.6, -0.6)$	$(-0.8, -0.6)$
MR_2	$(-0.8, 0.2)$	$(0.0, -0.6)$
MR_3	$(0.8, 0.0)$	$(-0.6, 0.6)$

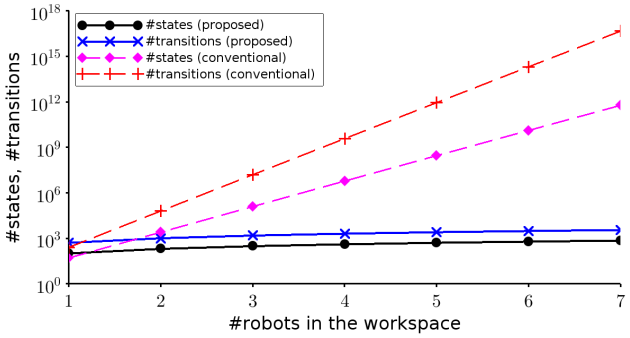

FIGURE 10. The numbers of states and transitions in the entire system.

In contrast, because of the introduction of the AS-Shields, the number of states in the entire system is reduced to $51 \times 2 \times N = 102N$ and that of transitions is to $51 \times 5 \times 2 \times N = 510N$ because each robot has its own original controller and AS-Shield.

B. AS-SHIELD WITHOUT DEADLOCK AVOIDANCE

We executed a simulation using the AS-Shield $\hat{S}_S^\alpha$ given by (34). The results are shown in Fig. 11. Note that there are no collisions among the robots, which is assured by Theorem 3. Each robot sends its current and next positions, and the shield checks the received position information. However, MR_2 and MR_3 do not arrive at their destinations because the shields in both MR_2 and MR_3 overwrite control inputs with $\hat{u}_\epsilon$. The current position of MR_2 is $(0.0, -0.2)$, and the next position is $(0.0, -0.4)$. The current position of MR_3 is $(0.0, -0.6)$, and the next position is $(0.0, -0.4)$, which is the same as MR_2. Consequently, a deadlock occurred.

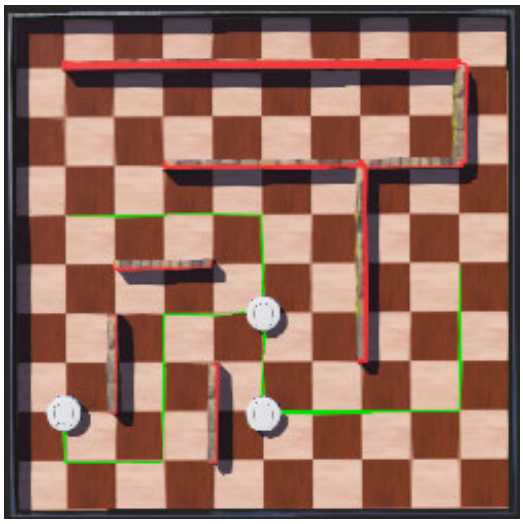

FIGURE 11. The deadlock by the collision avoidance.

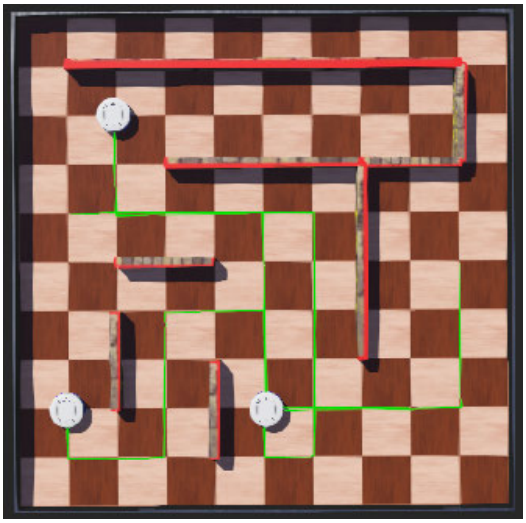

FIGURE 12. The successful behavior by the proposed controller.

C. AS-SHIELD WITH DEADLOCK AVOIDANCE

We executed a simulation using AS-Shield $\hat{S}_S^{\beta}$ given by (38). Each robot sends its current and next positions and a set of blocked positions, and the shield checks the received position information. The results are shown in Fig. 12. Each robot successfully arrived at its destination without collision or deadlock. When MR_2 and MR_3 arrived at the same positions as shown in Fig. 11, a possible deadlock is detected by MR_2 and MR_3. Then, MR_2 and MR_3 generate AS-Shield (38). In this experiment, MR_3 obtained an avoidance trajectory heading to the state $(0.0, -0.8, \pi)$, and MR_2 did not generate an avoidance trajectory. Note that MR_2 and MR_3 confirmed safety by communicating their positions on the avoidance trajectory (i.e., blocked positions). When MR_3 transits from the state $(0.0, -0.6, \pi/2)$ to $(0.0, -0.8, \pi/2)$, it blocks $(0.0, -0.6)$ and $(0.0, -0.8)$. When MR_3 proceeds to $(0.0, -0.8, \pi/2)$, the blocked position is only $(0.0, -0.8)$, and the next position of MR_3 is $(0.0, -0.8)$ because it transits from $(0.0, -0.8, \pi/2)$ to $(0.0, -0.8, \pi)$. Then, MR_2 proceeds to the position $(0.0, -0.6)$ and arrives at the destination. On the other hand, MR_3 transits from $(0.0, -0.8, \pi)$ to $(0.0, -0.8, \pi/2)$ for the shortest path from $(0.0, -0.8, \pi)$ to the target position $(-0.6, 0.6)$. However, the position $(0.0, -0.6)$ is occupied by MR_2. Then, MR_3 generates another avoidance trajectory $(0.0, -0.8, \pi/2) \rightarrow (0.0, -0.8, 0) \rightarrow (0.2, -0.8, 0) \rightarrow (0.2, -0.8, \pi/2) \rightarrow (0.2, -0.6, \pi/2) \rightarrow (0.2, -0.4, \pi/2)$. These positions are blocked while MR_3 is on the avoidance trajectory. When MR_3 arrives at $(0.2, -0.4, \pi/2)$, it resumes to head to the target position and finally arrives at $(-0.6, 0.6)$. Therefore, it is confirmed that each robot successfully arrives at its target state under unpredictable packet dropouts by dynamically generating avoidance trajectories with probabilistic computation and by negotiating the trajectories with the other robots.

TABLE 3. Simulation parameters ($N = 4$).

Robot	Initial position	Target position
MR_1	$(0.6, -0.6)$	$(-0.8, -0.6)$
MR_2	$(-0.8, 0.2)$	$(0.0, -0.6)$
MR_3	$(0.8, 0.0)$	$(-0.6, 0.6)$
MR_4	$(0.4, 0.6)$	$(0.8, 0.2)$

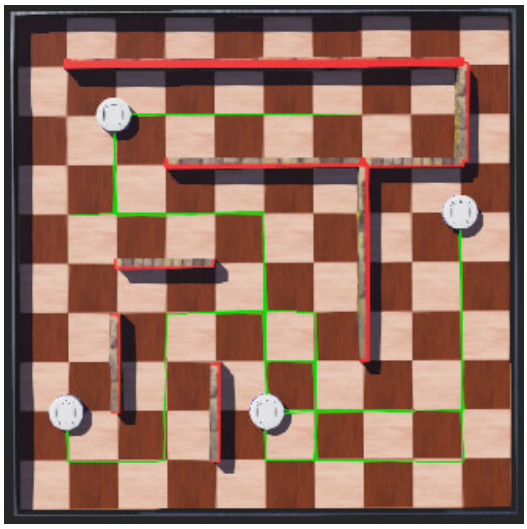

FIGURE 13. The successful behavior in spite of the different number of robots.

Let us consider the case where another robot MR_4 exists in the workspace. The initial and target positions are listed in Table 3. Recall that we designed homogeneous controllers. The controller for MR_4 is then obtained in the same manner as MR_1, MR_2, and MR_3. Figure 13 shows the results, and it is confirmed that the desired behavior was obtained.

VI. CONCLUSION

We propose alternating simulation-based shields (AS-Shields) attached to an abstraction-based controller. Because AS-Shields are constructed with symbolic control barrier functions such that transitions to unsafe states are disabled, it is theoretically assured that shield-attached abstraction-based controllers enforce both control and safety

specifications. We applied AS-Shields to a system consisting of multiple mobile robots sharing a workspace. We designed an AS-Shield for a system that achieves collision and deadlock avoidance under unpredictable packet dropouts among robots. We confirmed that all the robots arrived safely at their target states. In terms of the performance of mobile robots, several path-planning methods must be compared by investigating the effects of updating control inputs using AS-Shields. Future work should investigate the generality of the proposed method by applying it to other systems, such as autonomous cars, construction machineries, and unmanned aerial vehicles.

APPENDIX A
PROOF OF LEMMA 1

We prove that R_c defined by (17) satisfies the conditions for ASR from S_c to S_p.

First, consider any $x_{c0} \in X_{c0}$. Because $X_{c0} \subseteq X_{p0}$ holds, we have $x_{c0} \in X_{p0}$. Thus, $(x_{c0}, x_{c0}) \in R_{cX}$ holds.

Next, consider any $(x_c, x_c) \in R_{cX}$ and any $u_c \in U_c(x_c)$. Because $S_c \subseteq S_p$ holds, we have $r_c(x_c, u_c) = r_p(x_c, u_c)$. Thus, we have $\forall x'_p \in r_p(x_c, u_c) : [x'_p \in r_c(x_c, u_c)] \wedge [(x'_p, x'_p) \in R_{cX}]$. □

APPENDIX B
PROOF OF LEMMA 2

We prove that $\hat{R}_\cup$ defined by (21) is an ASR from $\hat{S}_\cup$ to $\hat{S}$ and that (9) is satisfied.

First, consider any $\hat{x}_{\cup 0} \in \hat{X}_{\cup 0}$. Since $\hat{X}_{\cup 0} = \hat{X}_0 \cap \hat{\mathcal{X}}$, we have $\hat{x}_{\cup 0} \in \hat{X}_0$ and $\hat{B}(\hat{x}_{\cup 0}) \geq 0$. Consequently, we have $(\hat{x}_{\cup 0}, \hat{x}_{\cup 0}) \in \hat{R}_{\cup X}$.

Next, consider any $(\hat{x}_\cup, \hat{x}) \in \hat{R}_{\cup X}$ such that $\hat{x}_\cup = \hat{x} \in \hat{\mathcal{X}}$ and any $\hat{u}_\cup \in \hat{U}_\cup(\hat{x}_\cup)$. Here, we present two cases.

1) Suppose $\hat{r}_C(\hat{x}_\cup, \hat{u}_\cup) \cap \hat{r}_S^m(\hat{x}_\cup, \hat{u}_\cup) \neq \emptyset$ holds. By (20), we have $\hat{u}_\cup \in \hat{\mathcal{U}}^{\hat{B}}(\hat{x}_\cup) = \hat{\mathcal{U}}^{\hat{B}}(\hat{x}) \subseteq \hat{U}(\hat{x})$. Then, we have $(\hat{x}_\cup, \hat{x}, \hat{u}_\cup, \hat{u}_\cup) \in \hat{R}_\cup$. In addition, for any $\hat{x}' \in \hat{r}(\hat{x}, \hat{u}_\cup)$, we have $\hat{x}'_\cup = \hat{x}' \in \hat{r}_C(\hat{x}_\cup, \hat{u}_\cup) \cap \hat{r}_S^m(\hat{x}_\cup, \hat{u}_\cup) \subseteq \hat{r}_\cup(\hat{x}_\cup, \hat{u}_\cup)$. Then, $(\hat{x}'_\cup, \hat{x}') \in \hat{R}_{\cup X}$ holds. Moreover, by the definition of $\hat{\mathcal{U}}^{\hat{B}}$, we have $\Delta\hat{B}(\hat{x}_\cup, \hat{u}_\cup) + \hat{B}(\hat{x}_\cup) \geq 0$. Thus, we have

$$\begin{aligned}\hat{B}(\hat{x}'_\cup) &\geq \min_{\hat{x}'' \in \hat{r}(\hat{x}_\cup, \hat{u}_\cup)} \hat{B}(\hat{x}'') \\ &= \Delta\hat{B}(\hat{x}_\cup, \hat{u}_\cup) + \hat{B}(\hat{x}_\cup) \geq 0, \end{aligned} \tag{40}$$

which implies $\hat{x}'_\cup = \hat{x}' \in \hat{\mathcal{X}}$.

2) Suppose $\hat{r}_C(\hat{x}_\cup, \hat{u}_\cup) \cap \hat{r}_S^m(\hat{x}_\cup, \hat{u}_\cup) = \emptyset$ and $\hat{r}_S(\hat{x}_\cup, \hat{u}_\cup) \neq \emptyset$ hold. By (16), we have $\hat{u}_\cup \in \hat{\mathcal{U}}^{\hat{B}}(\hat{x}_\cup) = \hat{\mathcal{U}}^{\hat{B}}(\hat{x}) \subseteq \hat{U}(\hat{x})$. Then, $(\hat{x}_\cup, \hat{x}, \hat{u}_\cup, \hat{u}_\cup) \in \hat{R}_\cup$ holds. Since $\hat{r}_S(\hat{x}_\cup, \hat{u}_\cup) = \hat{r}(\hat{x}, \hat{u}_\cup)$, for any $\hat{x}' \in \hat{r}(\hat{x}, \hat{u}_\cup)$, we have $\hat{x}'_\cup = \hat{x}' \in \hat{r}_S(\hat{x}_\cup, \hat{u}_\cup) \subseteq \hat{r}_\cup(\hat{x}_\cup, \hat{u}_\cup)$. Thus, $(\hat{x}'_\cup, \hat{x}') \in \hat{R}_{\cup X}$ holds. Moreover, by the definition of $\hat{\mathcal{U}}^{\hat{B}}$, we have $\Delta\hat{B}(\hat{x}_\cup, \hat{u}_\cup) + \hat{B}(\hat{x}_\cup) \geq 0$. Based on the same discussion as in the previous case, we have $\hat{x}'_\cup = \hat{x}' \in \hat{\mathcal{X}}$.

Therefore, the pair $(\hat{S}_\cup, \hat{R}_\cup)$ is a safe controller for $\hat{S}$ with respect to $\hat{B}$. □

APPENDIX C
PROOF OF THEOREM 3

By Lemma 2, the pair $(\hat{S}_\cup, \hat{R}_\cup)$ is a safe controller with respect to (10) for (2). In addition, the pair $(S_\cup, R_\cup)$ is the refined controller obtained by Theorem 1. Therefore, Theorem 2 shows that $(S_\cup, R_\cup)$ is a safe controller with respect to (8) for (1). □

REFERENCES

[1] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Princeton, NJ, USA: Princeton Univ. Press, 2009.

[2] W. Ren and R. W. Beard, *Distributed Consensus in Multi-Vehicle Cooperative Control: Theory and Applications*. Berlin, Germany: Springer, 2007.

[3] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Control barrier certificates for safe swarm behavior," in *Proc. IFAC Conf. Anal. Design Hybrid Syst.*, Oct. 2015, pp. 68–73.

[4] L. Wang, A. Ames, and M. Egerstedt, "Safety barrier certificates for heterogeneous multi-robot systems," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2016, pp. 5213–5218.

[5] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multiagent hybrid systems," *IEEE Trans. Autom. Control*, vol. 43, no. 4, pp. 509–521, Apr. 1998.

[6] M. Nagumo, "Über die Lage der integralkurven gewöhnlicher differentialgleichungen," in *Proc. Physico-Math. Soc. Jpn.*, vol. 24, 1942, pp. 551–559.

[7] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, Nov. 1999.

[8] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Proc. Int. Workshop Hybrid Syst., Comput. Control*, 1007, pp. 477–492.

[9] S. Prajna, "Barrier certificates for nonlinear model validation," *Automatica*, vol. 42, no. 1, pp. 117–126, Jan. 2006.

[10] P. Wieland and F. Allgower, "Constructive safety using control barrier functions," in *Proc. IFAC Symp. Nonlinear Control Syst.*, 2007, pp. 462–467.

[11] M. Z. Romdlony and B. Jayawardhana, "Uniting control Lyapunov and control barrier functions," in *Proc. 53rd IEEE Conf. Decis. Control*, Dec. 2014, pp. 2293–2298.

[12] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 3420–3431.

[13] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Proc. 53rd IEEE Conf. Decis. Control*, Dec. 2014, pp. 6271–6278.

[14] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.

[15] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation," in *Proc. Robot., Sci. Syst.*, vol. 13, 2017, pp. 1–10.

[16] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *Proc. Amer. Control Conf. (ACC)*, May 2021, pp. 3882–3889.

[17] R. Takano, H. Oyama, and M. Yamakita, "Application of robust control barrier function with stochastic disturbance model for discrete time systems," in *Proc. 5th IFAC Conf. Engine Powertrain Control, Simul. Model.*, vol. 51, 2018, pp. 46–51.

[18] Q. Hou and J. Dong, "Fully distributed cooperative fault-tolerant output regulation of linear heterogeneous MASs: A dynamic memory event-triggered distributed observer approach," *IEEE Trans. Autom. Sci. Eng.*, early access, Aug. 3, 2024, doi: 10.1109/TASE.2023.3298659.

[19] S. Yan, M. Shen, S. K. Nguang, G. Zhang, and L. Zhang, "A distributed delay method for event-triggered control of T–S fuzzy networked systems with transmission delay," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 10, pp. 1963–1973, Oct. 2019.

[20] M. Shen, X. Wu, J. H. Park, Y. Yi, and Y. Sun, "Iterative learning control of constrained systems with varying trial lengths under alignment condition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 9, pp. 6670–6676, Sep. 2023.

[21] M. Rungger and P. Tabuada, "A notion of robustness for cyber-physical systems," 2013, *arXiv:1310.5199*.
[22] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*, 1st ed. Berlin, Germany: Springer, 2009.
[23] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Berlin, Germany: Springer, 2009.
[24] R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi, "Alternating refinement relations," in *Proc. 9th Int. Conf. Concurrency Theory*, vol. 1466, 1998, pp. 163–178.
[25] A. Girard and G. J. Pappas, "Approximation metrics for discrete and continuous systems," *IEEE Trans. Autom. Control*, vol. 52, no. 5, pp. 782–798, May 2007.
[26] R. Goebel, R. G. Sanfelice, and A. R. Teel, *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton, NJ, USA: Princeton Univ. Press, 2012.
[27] A. Girard and G. J. Pappas, "Approximate bisimulation: A bridge between computer science and control theory," *Eur. J. Control*, vol. 17, nos. 5–6, pp. 568–578, Jan. 2011.
[28] G. Pola and P. Tabuada, "Symbolic models for nonlinear control systems: Alternating approximate bisimulations," *SIAM J. Control Optim.*, vol. 48, no. 2, pp. 719–733, Jan. 2009.
[29] M. Mizoguchi and T. Ushio, "Symbolic control of systems with dead times using symbolic Smith predictors," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Dec. 2016, pp. 5726–5731.
[30] M. Mizoguchi and T. Ushio, "Deadlock-free output feedback controller design based on approximately abstracted observers," *Nonlinear Anal., Hybrid Syst.*, vol. 30, pp. 58–71, Nov. 2018.
[31] M. Mizoguchi and T. Ushio, "Abstraction-based symbolic control barrier functions for safety-critical embedded systems," *IEEE Control Syst. Lett.*, vol. 6, pp. 1436–1441, 2022.
[32] R. Bloem, B. Könighofer, R. Könighofer, and C. Wang, "Shield synthesis," in *Proc. 21st Int. Conf. Tools Algorithms Construction Anal. Syst.*, vol. 9035, 2015, pp. 533–548.
[33] M. Wu, H. Zeng, and C. Wang, "Synthesizing runtime enforcer of safety properties under burst error," in *Proc. NASA Formal Methods Symp.*, 2016, pp. 65–81.
[34] S. Bharadwaj, R. Bloem, R. Dimitrova, B. Konighofer, and U. Topcu, "Synthesis of minimum-cost shields for multi-agent systems," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2019, pp. 1048–1055.
[35] J. Priya Inala, Y. Jason Ma, O. Bastani, X. Zhang, and A. Solar-Lezama, "Safe human-interactive control via shielding," 2021, *arXiv:2110.05440*.
[36] H. Hu, K. Nakamura, and J. F. Fisac, "SHARP: Shielding-aware robust planning for safe and efficient human–robot interaction," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 5591–5598, Apr. 2022.
[37] S. Pranger, B. Könighofer, M. Tappler, M. Deixelberger, N. Jansen, and R. Bloem, "Adaptive shielding under uncertainty," in *Proc. Amer. Control Conf. (ACC)*, May 2021, pp. 3467–3474.
[38] K. Kanashima and T. Ushio, "Finite-horizon shield for path planning ensuring safety/co-safety specifications and security policies," *IEEE Access*, vol. 11, pp. 11766–11780, 2023.
[39] K-Team, *Khepera IV*, K-Team Corp., Switzerland, 2014.
[40] Cyberbotics, *Webots—Open Source Robot Simulator*, Cyberbotics Ltd., Switzerland, 2018.

MASASHI MIZOGUCHI received the B.S., M.S., and Ph.D. degrees from Osaka University, in 2015, 2017, and 2023, respectively. He has been with Hitachi Ltd., since 2017. His current research interest includes abstraction-based formal synthesis and its implementation to real-time embedded systems. His research interests include software testing, hardware virtualization, real time scheduling for automotive software, and symbolic synthesis based on the approximate abstraction.

TOSHIMITSU USHIO (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Kobe University, in 1980, 1982, and 1985, respectively. He was a Research Assistant with the University of California at Berkeley, Berkeley, in 1985. From 1986 to 1990, he was a Research Associate with Kobe University, where he became a Lecturer, in 1990. He joined Osaka University as an Associate Professor, in 1994, became a Professor, in 1997, and became an Emeritus Professor, in 2023. His research interests include the control of discrete event systems and hybrid systems and the analysis of nonlinear systems.