

Discrete Abstractions of Nonlinear Systems Based on Error Propagation Analysis

Yuichi Tazaki, *Member, IEEE*, and Jun-ichi Imura, *Member, IEEE*

Abstract—This paper proposes a computational method for the feasibility check and design of discrete abstract models of nonlinear dynamical systems. First, it is shown that a given discrete-time dynamical system can be transformed into a finite automaton by embedding a quantizer into its state equation. Under this setting, a sufficient condition for approximate bisimulation in infinite steps of time between the concrete model and its discrete abstract model is derived. The condition takes the form of a set of linear inequalities and thus can be checked efficiently by a linear programming solver. Finally, the iterative refinement algorithm, which generates a discrete abstract model under a given error specification, is proposed. The algorithm is guaranteed to terminate in finite iterations.

Index Terms—Discrete abstraction, Nonlinear systems, Quantized systems

I. INTRODUCTION

In many control applications, controlled plants in the real world are modeled as systems with continuous (real-valued) states and dynamics expressed by ordinary differential equations or difference equations. In recent years, there have been growing interests towards analysis and control synthesis problems with complex specifications, apart from classical problem settings such as set-point regulation and trajectory tracking. However, analysis and control synthesis of complex systems require computational methods whose computational burden often become prohibitively large as the size of the system grows. In such cases, it is reasonable to make use of a simplified abstract model of the plant, which requires much less computational effort to analyze and to control, and still captures the essential characteristics of the plant. Among various kinds of abstract models, discrete abstract models, a class of systems whose state and control input take finitely many values, is of particular interest. This is because discrete abstract models enables us to treat various difficult properties of continuous-state systems: nonlinearity, discontinuity, and non-convexity in a uniform manner. In particular, some class of control problems are transformed into shortest-path problems on finite automata, which can be computed by well-known search algorithms whose complexities are of polynomial order; e.g., Dijkstra's algorithm and dynamic programming.

A. Brief Review of Existing Works

During this decade, there have been a variety of researches on discrete abstraction. Although these works share a common

objective, they vary in terms of: i) how they express discrete abstract models, and ii) what kind of applications they consider. In the following, we review some of the existing works and compare them mainly from these perspectives.

State Space Partitioning Approach

One intuitive way of deriving a discrete state system from a continuous state system is to aggregate neighboring states into a single symbolic state. More concretely, the state space is partitioned into finite regions, and each region is assigned a unique symbol. The major consideration here is i) how to partition the state space, and ii) how to define a transition relation over the symbolic states, so that the resultant symbolic system preserves the dynamical property of the original system.

In [1][2][3], quotient systems induced from the finite partitioning of state space have been investigated for different classes of systems. The studies focused on quotient systems with deterministic transitions, which are called finite bisimilarity quotients in [1]. Here, a partition-induced quotient system has a deterministic transitions if and only if the successor set of each region of the partition is included in a single region of the partition. The studies revealed that a partition that gives a finite bisimilarity quotient rarely exists, especially when one thinks of systems with continuous control inputs. Moreover, even if such a partition exists, it is often quite expensive to compute.

Some other researches address discrete abstraction by state space partitioning under milder conditions than bisimulations. Here, the symbolic transition is defined based on set intersection rather than set inclusion. This procedure results in a symbolic system with *non-deterministic* transitions, since the successor set of a single region may intersect with more than one regions. Using the terminology of bisimulation, such a symbolic system is similar but not bisimilar to the original system. This type of discrete abstract models is applied to safety verification problems in [4] and to control problems written in Linear Temporal Logic (LTL) formula in [20]. In [5], the non-determinism is treated as stochastic transitions. In [6], from a behavioral point of view, the set of all trajectories with a certain finite length is partitioned according to which regions and in what order they travel through. It has been shown that the behavior of a symbolic system obtained in this manner can be made arbitrarily close to that of the concrete system by considering longer trajectories.

State Waypoint Approach This approach [7][8] constructs a discrete abstract model by distributing a finitely many number of points called waypoints over the state space and assigning a symbolic state to each waypoint. A transition from one symbolic state to another one is assigned if the state of the concrete system is reachable from the waypoint corresponding to the former state to the waypoint corresponding to the

Y. Tazaki is with the Department of Mechanical Science and Engineering, Nagoya University, Nagoya, Japan. tazaki@nuem.nagoya-u.ac.jp

J. Imura is with the Department of Mechanical and Environmental Informatics, Tokyo Institute of Technology, Meguro, Tokyo, Japan. imura@mei.titech.ac.jp

latter state. This method has been applied to optimal control problems with non-convex state constraints. An interesting fact is that, although not explicitly mentioned in the works cited above, a discrete abstract model is constructed in such a way that the concrete system is *similar* to the abstract model in the bisimulation sense, as opposed to the methods based on state-space partitioning, which are mostly applied to reachability analysis. One critical shortcoming of this approach is that it requires the initial state to be set on one of the waypoints.

Approximate Bisimulation Approach

There are some disadvantages in discrete abstractions based on simulation relations, which is due to the fact that simulation is *one-sided*. Recently, some attempts have been made to address discrete abstraction based on the notion of approximate bisimulation [10]. Approximate bisimulation is an extension of the original bisimulation [9] to the metric space. It admits an equivalence relation between two systems if the distance of output signals can be kept within a given threshold. From a geometric point of view, an approximate bisimulation relation between a discrete abstract model and a concrete system forms a covering of the continuous state space, which is a fundamental difference from partitioning-based methods. Until now, it has been shown that discrete abstract models of a wide range of systems can be obtained based on approximate bisimulation [11][14][15].

Approximate bisimulation-based approach is especially suitable for control problems with a quantitative performance measure. It also has an advantage that it does not require expensive geometric computations to construct a discrete abstract model. To date, however, it has the following limitations. First, the conventional approximate bisimulation requires constant precision; that is, it requires the distance of output variables to be below a specified constant, independent of the values of output variables. In practice, it is much more desirable if the precision can be varied according to the signal value. Second, the distribution of discrete states is also uniform, meaning that the number of discrete states grows exponentially with respect to the dimension of the continuous state space. As the third point, it requires the global incremental stability. Many nonlinear systems could have locally incrementally unstable regions, so this point may severely limit the range of application.

B. Contribution and Organization of This Paper

Motivated by the above backgrounds, this paper proposes a method for discrete abstraction of nonlinear systems using variable-resolution quantizers, which is capable of handling error specifications with higher degrees of freedom. The organization and concrete contributions of the paper are summarized below.

Approximate Bisimulation with Higher Degrees of Freedom in Error Specification (Section II)

First, the notion of approximate bisimulation is extended to cover time-varying and signal-dependent error specifications. The extended framework includes the conventional approximate bisimulation with uniform error conditions and the finite-step variant as its subclasses.

Expression of Discrete Abstract Models by Means of Quantizer Embedding (Section III)

An expression of discrete abstract models based on the quantization of state variable and control input is proposed. Quantization of input/output signals has already been considered in the literature as a way to obtain a discrete abstract model. However, the term *quantization* in the existing works refers to a map that assigns a symbol according to which region of a partition the signal is included in. As previously mentioned, discrete models obtained by this type of quantization exhibit non-deterministic transitions, even if the concrete system is deterministic. This unnecessary non-determinism causes conservativeness in verification and controller design. In this paper, we consider a different type of quantization, which resets the signal at every time instant to the representative value of a region in which the signal value is included. This reset-type quantizer is directly embedded in the state equation of the concrete system to transform it into a discrete model with deterministic transitions. Moreover, by expressing quantizers in terms of variable-resolution meshes, one can naturally create discrete abstract models in which the quantized values of states and inputs are non-uniformly distributed over the continuous state space and the input space.

Conditions for Approximate Bisimulation Based on Error Propagation Analysis (Section IV)

As previously mentioned, conventional approximate bisimulation requires constant precision. To cope with approximate bisimulation in a generalized setting, a fundamentally new analytical framework is needed. In the proposed framework, an approximate bisimulation relation is expressed as an *error distribution*, a collection of error variables. Each error variable is assigned to a state of the discrete abstract model and it represents the acceptable distance between that discrete state and the state of the concrete model. It is shown that approximate bisimulation is achieved when the error distribution satisfies a certain stationarity condition with respect to the transition relation of the discrete model. This framework allows time-varying, signal-dependent error specifications as well as concrete system models that are not globally incrementally stable. Moreover, the obtained condition can be checked by solving a single linear program (LP).

Computational Method for the Feasibility Check and Design of Discrete Abstract Models (Section V)

An algorithm that repeatedly refines a variable-resolution mesh until approximate bisimulation is achieved, is presented. The algorithm makes use of the information of *active constraints* of LP to determine which portion of the mesh to refine. The proposed algorithm is guaranteed to terminate after finite iterations and to output a discrete abstract model for a given concrete system satisfying a certain existence condition.

Application of Discrete Abstractions to Finite-Horizon Optimal Control (Section VI-C)

There exist a number of works in which discrete abstraction has been applied to control problems (see, for example, [17], [13], [18], [19] and [21]). Control is often done in a hierarchical manner, in which a rough trajectory is planned using a discrete abstract model, and then the concrete system is driven by a feedback controller to track the planned

symbolic trajectory. However, the relation between the level of sub-optimality and approximation precision has not been thoroughly discussed. Although control laws based on abstract models are fundamentally approximate, it is still important to have a quantitative measure on how close the obtained approximate controller is to the optimal controller. In this paper, it is shown that using an approximately bisimilar abstraction, one can construct a sub-optimal solution to finite-horizon optimal control problems provided with a known amount of performance deterioration parametrized by the approximation precision. This application also highlights the usefulness of finite-step bisimulation.

Finally, concluding remarks are given in Section VII.

Notation: We write $[i_1 : i_2]$ to express the sequence of integers $i_1, i_1 + 1, \dots, i_2$. The symbol $\mathbb{R}$ denotes the field of real numbers and the symbol $\mathbb{Z}^+$ denotes the set of non-negative integers. For a function f and a set X , we define $f(X) = \{f(x) | x \in X\}$. Similar notation is used for multivariate functions. For two sets X and Y , $X + Y$ denotes the Minkowski sum; $X + Y = \{x + y | x \in X, y \in Y\}$.

II. APPROXIMATE BISIMULATION OF DISCRETE-TIME SYSTEMS

A. System Description

In this paper, we address the discrete abstraction of a class of discrete-time systems governed by the following difference equation:

$$x_{t+1} = f(x_t, u_t). \quad (1)$$

Here, $x_t \in \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$ are the state and control input of the system at time $t \in \mathbb{Z}^+$, respectively, and $f : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$ is the state transition function. We assume that f is smooth. In order to approximate the behavior of a system of this class using finite symbolic states and inputs, we need to specify a bounded subset of the system's behaviors that is to be approximated. Let us define the target behavior set as a set of trajectories satisfying

$$x_0 \in X_0, x_t \in X, u_t \in U, x_{t+1} = f(x_t, u_t) \ (t \geq 0). \quad (2)$$

Here, $X \subset \mathbb{R}^n$ and $U \subset \mathbb{R}^m$ specify the target region of approximation and $X_0 \subseteq X$ is the set of initial states. As described later, a trajectory that approximates another trajectory included in the target behavior set may exceed X and U , as long as it satisfies a certain error condition. In this sense, the sets X and U are solely for specifying the target region of abstraction and they do not impose any constraint on the system's trajectories.

Based on the above discussion, a discrete-time system is formally defined as follows:

Definition 1 Discrete-time dynamical system

A discrete-time system is a tuple $\langle X, X_0, U, f \rangle$ whose behavior is governed by (1). The sets $X \subset \mathbb{R}^n$, $X_0 \subseteq X$ and $U \subset \mathbb{R}^m$ define the target behavior set as given by (2).

Remark 1 As we begin with discrete-time systems, the discretization of time for continuous-time systems is beyond the scope of this paper; we focus on the discretization of states and control inputs.

B. Time-Varying Approximate Bisimulation

In the following, we introduce the notion of time-varying approximate bisimulation as a generalization of the original approximate bisimulation, which was first proposed by Girard and Pappas [10] as a framework to compare two systems in a quantitative manner. It admits equivalence relation between two systems if the distance of output signals can be maintained below a certain constant for infinite steps of time. On the other hand, a finite-step variant of approximate bisimulation was considered in [24]. Although it guarantees approximate bisimulation for only a finite duration of time, it has the following advantages over the infinite-step type. First, discrete abstract models based on finite-step approximate bisimulation can be obtained by discretization with lower resolutions compared to those based on the infinite-step type. Indeed, the smaller the number of time steps, the lower resolution required. Lower resolution implies less states and input values in discrete abstract models and thus helps to save memory consumption and to reduce the computation costs of various symbolic algorithms. One typical application of finite-step bisimulation is model predictive control. To compute an approximate solution to finite-horizon optimal control problems, which must be solved at every control cycle of MPC, one may use a discrete abstract model that guarantees an error specification only in the prediction horizon. Second, while infinite-step discrete abstract models do not always exist, finite-step models are necessarily available under very mild conditions.

To take advantage of both types, we consider a hybrid formulation, which includes both infinite-step and finite-step approximate bisimulations as special cases. First, we introduce the following notation:

Definition 2 Let $\Sigma = \langle X, X_0, U, f \rangle$ and $\hat{\Sigma} = \langle \hat{X}, \hat{X}_0, \hat{U}, \hat{f} \rangle$ be discrete-time dynamical systems. Moreover, let $R^x \subset \mathbb{R}^n \times \mathbb{R}^n$ and $R^{x'} \subset \mathbb{R}^n \times \mathbb{R}^n$ be binary relations of states and $R^u \subset \mathbb{R}^m \times \mathbb{R}^m$ be a binary relation of control inputs. We write

$$R^x \lesssim_{R^u} R^{x'} \quad (3)$$

if and only if for any $(x, \hat{x}) \in R^x$, the following holds: for any $u \in U$ satisfying $f(x, u) \in X$, there exists a $\hat{u} \in \mathbb{R}^m$ satisfying

$$(u, \hat{u}) \in R^u, (f(x, u), \hat{f}(\hat{x}, \hat{u})) \in R^{x'}, \quad (4)$$

and for any $\hat{u} \in \hat{U}$ satisfying $\hat{f}(\hat{x}, \hat{u}) \in \hat{X}$, there exists a $u \in \mathbb{R}^m$ satisfying (4).

The explanation of the above notation is as follows. Suppose the states of Σ and $\hat{\Sigma}$ are in the relation R^x at some time instant. If the system Σ makes a transition with a control input chosen from U and moves to a state inside X , then the system $\hat{\Sigma}$ should be able to make a transition to the next state while satisfying the relation R^u on the control inputs and the relation $R^{x'}$ on the states at the next time instant, and vice versa.

Using the above notation, the definition of bisimulation used in this paper is given below.

Definition 3 *Time-Varying Bisimulation*

Let $\Sigma = \langle X, X_0, U, f \rangle$ and $\hat{\Sigma} = \langle \hat{X}, \hat{X}_0, \hat{U}, \hat{f} \rangle$ be discrete-time dynamical systems. A set of binary relations

$$R = (\{R_t^x\}_{0:N}, \{R_t^u\}_{0:N}), \quad (5)$$

$$(R_t^x \subset \mathbb{R}^n \times \mathbb{R}^n, R_t^u \subset \mathbb{R}^m \times \mathbb{R}^m)$$

is a time-varying bisimulation if and only if the following holds:

1) for any $x \in X_0$, there exists a $\hat{x} \in \hat{X}_0$ such that

$$(x, \hat{x}) \in R_0^x \quad (6)$$

holds, and for any $\hat{x} \in \hat{X}_0$, there exists a $x \in X_0$ such that (6) holds.

$$2) \quad R_t^x \preceq_{R_t^u} R_{t+1}^x \text{ for all } t \in [0 : N-1], \quad (7a)$$

$$R_N^x \preceq_{R_N^u} R_N^x. \quad (7b)$$

In Definition 3, Condition 1 requires that for any initial state of one system, there should exist an initial state of the other system satisfying the relation R_0^x . In Condition 2, (7a) requires that the relations R_t^x and R_t^u , which are time varying, should be maintained for the first N steps. Moreover, (7b) requires that R_N^x should be maintained indefinitely for $t \geq N$. Note that if N is set as 0, then the above definition becomes infinite-step but stationary bisimulation. If $N > 0$ and (7b) is dropped, it becomes finite-step bisimulation.

Now, let

$$\bar{R} = (\{\bar{R}_t^x\}_{0:N}, \{\bar{R}_t^u\}_{0:N}), \quad (8)$$

$$(\bar{R}_t^x \subset \mathbb{R}^n \times \mathbb{R}^n, \bar{R}_t^u \subset \mathbb{R}^m \times \mathbb{R}^m)$$

be another set of binary relations. We call $\bar{R}$ an *error specification*. We say $R = (\{R_t^x\}_{0:N}, \{R_t^u\}_{0:N})$ is a time-varying approximate bisimulation satisfying the error specification $\bar{R} = (\{\bar{R}_t^x\}_{0:N}, \{\bar{R}_t^u\}_{0:N})$ if $R_t^x \subseteq \bar{R}_t^x$, $R_t^u \subseteq \bar{R}_t^u$ hold for all $t \in [0 : N]$.

A simple example of $\bar{R}_t^x$ is a uniform error condition: $\bar{R}_t^x = \{(x, \hat{x}) \mid \|x - \hat{x}\| \leq \epsilon\}$ where ϵ is a positive constant. Only this type of error condition has been considered in the conventional approximate bisimulation. On the other hand, $\bar{R}_t^x$ defined as $\bar{R}_t^x = \{(x, \hat{x}) \mid \|x - \hat{x}\| \leq \eta \min\{\|x\|, \|\hat{x}\|\} + \epsilon\}$ where both ϵ and η are positive constants, expresses a relative error condition. As shown above, our definition of bisimulation admits large flexibility in the design of error specification.

Definition 4 *Discrete-time systems Σ and $\hat{\Sigma}$ are approximately bisimilar with respect to an error specification $\bar{R}$ if and only if there exists a time-varying approximate bisimulation R that satisfies $\bar{R}$.*

III. QUANTIZER EMBEDDING

In this section, we introduce the notion of *quantizer embedding*, which is a transformation from a continuous-state system to a finite automaton.

A. Finite automata induced by quantizer-embedding

First of all, we introduce *mesh* structure defined in a bounded region in $\mathbb{R}^n$. A mesh M is a finite collection of pairs of a node and a cell: $M = \{(C_s, \xi_s) \mid s \in \mathcal{S}\}$. Here, $\mathcal{S}$ denotes a finite set of identifiers. Each C_s ($s \in \mathcal{S}$) is a compact subset of $\mathbb{R}^n$ and called a *cell* of the mesh. The cells are mutually disjoint; $C_{s_1} \cap C_{s_2} = \emptyset$ ($s_1 \neq s_2$). A region covered by the cells is called the *domain* of the mesh and denoted by $\text{dom}(M) = \bigcup_s C_s$. Each $\xi_s \in C_s$ is called the *node* of the cell C_s . Moreover, we call a subset of M a *sub-mesh* of M .

A mesh M defines a quantization function on $\mathbb{R}^n$ as shown below.

$$Q[M] : \text{dom}(M) \mapsto \{\xi_s\}_{s \in \mathcal{S}}, \quad Q[M](x) = \xi_s \text{ if } x \in C_s. \quad (9)$$

A quantization function $Q[M](\cdot)$ maps an arbitrary point $x \in \text{dom}(M)$ to a node whose corresponding cell includes x .

Now, for a system $\Sigma = \langle X, X_0, U, f \rangle$, we define mesh structures in the state space and the input space as $M^x = \{(C_s, \xi_s) \mid s \in \mathcal{S}\}$, $M^u = \{(C_a, \xi_a) \mid a \in \mathcal{A}\}$, respectively. We call M^x the state mesh and M^u the input mesh. The symbol $\mathcal{S}$ denotes the set of cell identifiers of the state mesh M^x and $\mathcal{A}$ denotes that of the input mesh M^u . Moreover, we use the subscript s for the cells and nodes of the state mesh and the subscript a for those in the input mesh. Accordingly, let us call $Q^x(\cdot) = Q[M^x](\cdot)$ the state quantizer and $Q^u(\cdot) = Q[M^u](\cdot)$ the input quantizer. It is required that the domain of the state mesh and the domain of the input mesh, $\text{dom}(M^x)$ and $\text{dom}(M^u)$, include X and U , respectively, with certain margins. More detailed conditions will be given later. From now on, we call a pair of a state mesh and an input mesh, $\mathcal{M} = (M^x, M^u)$, simply a mesh.

We introduce the following notation and terminology for later use. For a compact set S , Let $\underline{r}(S, p)$ denote the maximum radius of a ball centered at p included in S . Similarly, let $\bar{r}(S, p)$ be the minimum radius of a ball centered at p that includes S . For a cell (C, ξ) , we call $\bar{r}(C, \xi)$ the *radius* of the cell. Moreover, the *resolution* of a mesh $\mathcal{M}$ is defined as its largest cell radius.

A discrete-time system is transformed into a finite automaton by embedding a quantizer into its state-transition function.

Definition 5 *Quantizer embedding of discrete-time systems*

Let $\Sigma = \langle X, X_0, U, f \rangle$ be a discrete-time system. Moreover, let $\mathcal{M} = (M^x, M^u)$ be a mesh. The quantizer embedding of Σ , denoted by $\text{QE}(\Sigma, \mathcal{M})$, is a system $\hat{\Sigma} = \langle Q^x(X), Q^x(X_0), Q^u(U), \hat{f} \rangle$ whose state transition function is defined as

$$\hat{f}(\hat{x}, \hat{u}) = Q^x(f(\hat{x}, \hat{u})) \quad (10)$$

where $\hat{x}$ and $\hat{u}$ are the state and input of $\hat{\Sigma}$, respectively.

The control input of $\hat{\Sigma}$ is chosen from the nodes of M^u . As illustrated in Fig. 1, at every transition, the state of $\hat{\Sigma}$ is reset to the node of a cell of M^x in which the state immediately after the transition made by f is included. Since both M^x and M^u are composed of a finite number of cells, a system with

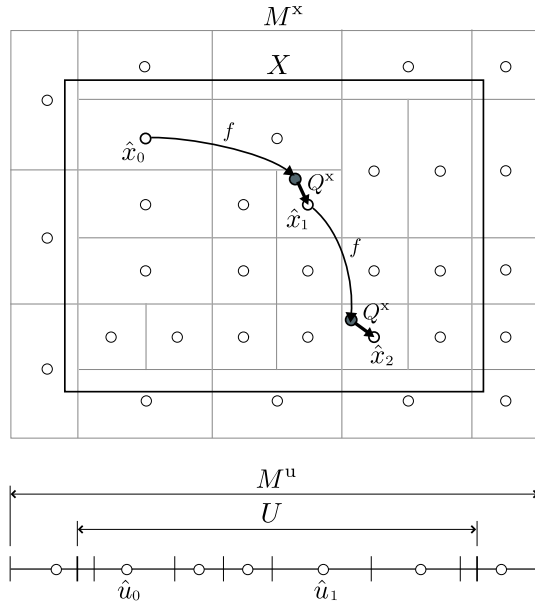


Fig. 1. Transition of a quantizer-embedded system.

a state-transition of the form (10) can be viewed as a finite automaton. Once we assume that a discrete abstract model is obtained by means of the quantizer embedding of the concrete system, the design of a discrete abstract model reduces to the design of a mesh. The transition of $\hat{\Sigma}$ can be rewritten in a symbolic form as

$$s \xrightarrow{a} s' \Leftrightarrow Q^x(f(\xi_s, \xi_a)) = \xi_{s'}. \quad (11)$$

The indices of state cells and inputs cells, s and a , are now regarded as the symbolic state and symbolic input of $\hat{\Sigma}$, respectively. Accordingly, $\mathcal{S}$ and $\mathcal{A}$ denote the set of symbolic states and that of symbolic inputs, respectively.

In order to compute which state s' a pair (s, a) moves to, one must compute which cell $C_{s'}$ includes $f(\xi_s, \xi_a)$. The computational complexity of identifying a cell including a given point depends on the class of meshes being used. For hierarchically partitioned mesh, which we will introduce later in Section V-B, this computation can be done quite efficiently by utilizing its tree structure. For later use, we define the following symbolic sets:

$$\begin{aligned} \mathcal{S}_0 &= \{s \mid \xi_s \in X_0\}, \\ \mathcal{S}_t &= \{s' \mid s \xrightarrow{a} s', s \in \mathcal{S}_{t-1}, a \in \mathcal{A}\}, \\ \mathcal{S}_{N+} &= \bigcup_{t \geq N} \mathcal{S}_t, \\ \hat{\mathcal{S}}_t &= \begin{cases} \mathcal{S}_t & \text{if } t \in [0 : N-1], \\ \mathcal{S}_{N+} & \text{if } t = N. \end{cases} \end{aligned} \quad (12)$$

Here, $\mathcal{S}_t$ is defined recursively as a set of symbolic states that $\hat{\Sigma}$ may take at time t given $\mathcal{S}_{t-1}$, and $\mathcal{S}_{N+}$ is a set of symbolic states that $\hat{\Sigma}$ may take at time N and beyond. The symbol $\hat{\mathcal{S}}_t$

is introduced solely for convenience. Furthermore, we write

$$\begin{aligned} (t, s) \xrightarrow{a} (t', s') &\Leftrightarrow \\ (t \in [0 : N-1], t' = t+1, s \in \mathcal{S}_t \text{ and } s \xrightarrow{a} s') &\text{ or } \\ (t = t' = N, s \in \mathcal{S}_{N+} \text{ and } s \xrightarrow{a} s'). \end{aligned} \quad (13)$$

One can consider two other forms of quantizer embedding. The first form embeds a state-quantizer alone and leaves control inputs not quantized. In this case, the symbolic state transition implicitly defines a partition in the input space: $U_{s,s'} := \{u \in U \mid Q^x(f(\xi_s, u)) = \xi_{s'}\}$. For each s , $U_{s,s'}$ ($s' \in \mathcal{S}$) form a partition of U (thus the partition is state dependent). The second form defines a quantizer in the product space of states and inputs. Let $Q^{xu}(\cdot) = Q[M^{xu}](\cdot)$ be a quantizer defined in $X \times U$. Each node of M^{xu} represents a symbolic state of the quantizer embedding. In this case, the input space is implicitly partitioned in the following way: $U_{s,s'} := \{u \in U \mid Q^{xu}(f(\xi_s, u)) = \xi_{s'}\}$ (note here that $\xi_s \in X \times U$ in this setting). Although both formulations are of theoretical interest, they have a common shortcoming from a practical perspective; that is, the computation of the symbolic transition relation is expensive. In the above formulations, one needs to solve a small inverse problem to check if there exists a transition for a tuple (s, a, s') . In the case of a linear system and a mesh with polytopic cells, it amounts to a single linear program. Although each of them is quite small, as the number of cells grows, the total computation cost becomes significant. On the other hand, with separate meshes M^x and M^u , the computation of symbolic transitions is quite cheap; for each pair (s, a) , one simply computes which cell the value $f(\xi_s, \xi_a)$ falls into. For this reason, we employ the quantization embedding of this type.

Based on the discussion made so far, we formulate the problem of discrete abstraction as follows.

Problem 1 Discrete Abstraction

For a discrete-time system $\Sigma = \langle X, X_0, U, f \rangle$ and an error specification $\bar{R} = (\{\bar{R}_t^x\}_{0:N}, \{\bar{R}_t^u\}_{0:N})$, find a mesh $\mathcal{M} = (M^x, M^u)$ such that the quantizer-embedding $\text{QE}(\Sigma, \mathcal{M})$ is approximately bisimilar to Σ with respect to $\bar{R}$.

IV. CONDITIONS FOR APPROXIMATE BISIMULATION

The goal of this section is to derive a sufficient condition for time-varying approximate bisimulation that is computationally inexpensive to check. The first step is to decompose the approximate bisimulation relation with respect to the states and inputs of the discrete abstract model. For preparation, let us introduce the following notation:

$$\begin{aligned} R_{t,s} &= \{x \mid (x, \xi_s) \in R_t^x\}, & R_{t,a} &= \{u \mid (u, \xi_a) \in R_t^u\}, \\ \bar{R}_{t,s} &= \{x \mid (x, \xi_s) \in \bar{R}_t^x\}, & \bar{R}_{t,a} &= \{u \mid (u, \xi_a) \in \bar{R}_t^u\}. \end{aligned} \quad (14)$$

Now, observe that some existence quantifiers appear in the definition of approximate bisimulation, which are quite difficult to handle directly in the design of discrete abstract models. To transform them into tractable forms, we replace these quantifiers with explicit maps. We call these maps *interfaces*, borrowing the term from [13]. For Condition 1 in

Definition 3, we consider here the case that the initial state of one system is given by an explicit function of the initial state of the other system; that is, given an initial state x_0 of Σ , the initial state of $\hat{\Sigma}$ is given by

$$\hat{x}_0 = Q^x(x_0) \quad (15)$$

where Q^x is the same state quantizer used to define the discrete abstract model. In the opposite case, the initial state of Σ is set as

$$x_0 = \hat{x}_0. \quad (16)$$

Similarly, for Condition 2, we consider the case that the simulating input of one system is given by an explicit function of the states of both systems and the input of the other system. Given x_t , $\hat{x}_t$ and u_t ,

$$\hat{u}_t = Q^u(u_t + \phi(\hat{x}_t, x_t)), \quad (17)$$

and in the opposite case,

$$u_t = \hat{u}_t + \phi(x_t, \hat{x}_t) \quad (18)$$

where Q^u is the same input quantizer used to define the discrete abstract model and ϕ is a smooth function satisfying $\phi(x, \hat{x}) = -\phi(\hat{x}, x)$. We call ϕ the *input interface*. Although more general forms of input interface may be considered, we restrict our attention to this form to express conditions for bisimulation in a compact form, as we will see later. The simplest input interface is the common input type; $\phi(x, \hat{x}) = 0$. In many cases, however, one can use more sophisticated input interfaces to satisfy a given error specification with coarser meshes. In fact, the essential role of input interface is to improve the stability of the error dynamics. Suppose no quantization is applied to $\hat{\Sigma}$, and let us write $\delta x = \hat{x} - x$, $\delta u = \hat{u} - u$. Then the error dynamics is given by

$$\begin{aligned} \delta x_{t+1} &= f(\hat{x}_t, \hat{u}_t) - f(x_t, u_t) \approx \frac{\partial f}{\partial x} \delta x_t + \frac{\partial f}{\partial u} \delta u_t \\ &\approx \left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial u} \frac{\partial \phi}{\partial x} \right) \delta x_t. \end{aligned}$$

In the above equation, we see that the stability of error dynamics is affected by the choice of ϕ . For linear systems given by $x_{t+1} = Ax_t + Bu_t$, it is natural to use a linear input interface of the form $\phi(x, \hat{x}) = F(x - \hat{x})$. In this way, one can construct a discrete abstract model under infinite-step bisimulation even if the system itself is unstable, by selecting F so that $A + BF$ is stable. In [22], the simultaneous design of the matrix F in the input interface and a mesh is addressed using an LMI-based technique. For nonlinear systems, however, there is no general methodology for designing ϕ , just like there is no general solution for global stabilization of nonlinear systems. In such cases, one should exploit the property of each particular system to design a proper input interface. When finite-step bisimulation is considered or the system is inherently incrementally stable, one can always resort to the common-input type interface. In order to cover various situations considered above, in the following discussion, we assume that an input interface is given a priori.

Next, we introduce an assumption on the mesh domain.

Assumption 1 For any $(t, x, u, \hat{x}, \hat{u})$ such that $t \in [0 : N]$, $x \in X$, $u \in U$, $(u, \hat{u}) \in \bar{R}_t^u$ and $(x, \hat{x}) \in \bar{R}_t^x$,

$$\hat{u} \in \text{dom}(M^u), u + \phi(\hat{x}, x) \in \text{dom}(M^u) \quad (19)$$

hold. Moreover, for any $(t, x, \hat{x}, \hat{u})$ such that $t \in [0 : N]$, $x \in X$, $(x, \hat{x}) \in \bar{R}_t^x$ and $\hat{u} \in \text{dom}(M^u)$,

$$\hat{x} \in \text{dom}(M^x), f(\hat{x}, \hat{u}) \in \text{dom}(M^x) \quad (20)$$

hold.

Here, we assume that the state mesh and input mesh cover the target state set X and the target input set U , respectively, with enough margin.

Under Assumption 1, we obtain the following Lemma.

Lemma 1 Let Σ be a system, $\bar{R}$ be an error specification and ϕ be an input interface. Moreover, let $\mathcal{M}$ be a mesh satisfying Assumption 1. Furthermore, for (t, t') such that $t \in [0 : N - 1]$, $t' = t + 1$ or $t = t' = N$, consider binary relations $R_t^x \subseteq \bar{R}_t^x$, $R_t^u \subseteq \bar{R}_t^u$ and $R_{t'}^x \subseteq \bar{R}_{t'}^x$. Then, for Σ and $\hat{\Sigma} = \text{QE}(\Sigma, \mathcal{M})$, the relation $R_t^x \preceq_{R_t^u} R_{t'}^x$ holds if

$$\begin{aligned} f_s^\phi(R_{t,s}, C_a) &\subseteq R_{t',s'}, \\ \phi_s(R_{t,s}) + C_a &\subseteq R_{t,a} \end{aligned} \quad (21)$$

hold for all (s, a, s') such that $(t, s) \xrightarrow{a} (t', s')$, where $f_s^\phi(x, u) = f(x, u + \phi(x, \xi_s))$, $\phi_s(x) = \phi(x, \xi_s)$.

Proof: Take any x, u and s such that $x \in X$, $u \in U$, $f(x, u) \in X$, $s \in \hat{\Sigma}$ and $(x, \xi_s) \in R_t^x$. Let $u' = u + \phi(\xi_s, x)$. From (19), we have $u' \in \text{dom}(M^u)$, so let $\xi_a = Q^u(u')$. Moreover, from (20), $f(\xi_s, \xi_a) \in \text{dom}(M^x)$, so let $\xi_{s'} = Q^x(f(\xi_s, \xi_a))$. Since $x \in R_{t,s}$ and $u' \in C_a$, from (21) we have

$$\begin{aligned} f(x, u) &= f(x, u' + \phi(x, \xi_s)) = f_s^\phi(x, u') \in R_{t',s'}, \\ u &= u' + \phi(x, \xi_s) \in R_{t,a}, \end{aligned}$$

which is equivalent to $(f(x, u), \xi_{s'}) \in R_{t'}^x$, $(u, \xi_a) \in R_t^u$.

Next, take any s, a and x such that $s \in \hat{\Sigma}$, $\xi_s \in X$, $a \in \mathcal{A}$, $\xi_a \in U$ and $(x, \xi_s) \in R_t^x$. Again from (20), $f(\xi_s, \xi_a) \in \text{dom}(M^x)$, so let $\xi_{s'} = Q^x(f(\xi_s, \xi_a))$. Moreover, let $u = \xi_a + \phi(x, \xi_s)$. Since $x \in R_{t,s}$ and $\xi_a \in C_a$, we have

$$\begin{aligned} f(x, u) &= f(x, \xi_a + \phi(x, \xi_s)) = f_s^\phi(x, \xi_a) \in R_{t',s'}, \\ u &= \xi_a + \phi(x, \xi_s) \in R_{t,a}, \end{aligned}$$

and thus $(f(x, u), \xi_{s'}) \in R_{t'}^x$, $(u, \xi_a) \in R_t^u$. This concludes the proof. ■

In Lemma 1, thanks to the input interface, simulation conditions in both directions are expressed in a unified form. Moreover, since the discrete abstract model has only finite states and inputs, this condition takes the form of a finite collection of set inclusions. Based on Lemma 1, the next theorem transforms the original conditions of approximate bisimulation into a collection of set inclusions.

Theorem 1 Let Σ be a system, $\mathcal{M}$ be a mesh and $\bar{R}$ be an error specification. Moreover, let ϕ be an input interface. Then,

a set of binary relations $R = (\{R_t^x\}_{0:N}, \{R_t^u\}_{0:N})$ is a time-varying bisimulation between Σ and the quantizer-embedding $\hat{\Sigma} = \text{QE}(\Sigma, \mathcal{M})$ if the following conditions hold:

$$1') \text{ for all } s \in \mathcal{S}_0, \quad R_{0,s} \supseteq C_s. \quad (22)$$

2') for all (t, s, a, t', s') such that $(t, s) \xrightarrow{a} (t', s')$,

$$f_s^\phi(R_{t,s}, C_a) \subseteq R_{t',s'}, \quad (23a)$$

$$\phi_s(R_{t,s}) + C_a \subseteq R_{t,a}. \quad (23b)$$

Moreover, R satisfies $\bar{R}$ if and only if:

3') for all $t \in [0 : N]$, $s \in \hat{\mathcal{S}}_t$ and $a \in \mathcal{A}$,

$$R_{t,s} \subseteq \bar{R}_{t,s}, \quad R_{t,a} \subseteq \bar{R}_{t,a}. \quad (24)$$

Proof: Using the initial-state interface, condition 1 in Definition 3 is expressed as: $(x, Q^x(x)) \in R_0^x$ for any $x \in X_0$, which is equivalent to condition 1'. Next, from Lemma 1, we can see that condition 2' implies condition 2. Finally, condition 3' is obviously equivalent to condition 3. ■

The first statement in this theorem is sufficient but not necessary. This is because existence quantifiers in the definition of approximate bisimulation are replaced by interfaces. At the price of this conservatism, the theorem provides us with a set of conditions in a tractable form. However, the above conditions involve set inclusions with nonlinear transformations, which are computationally quite expensive to check. Moreover, finding a collection of sets that satisfies the inclusion conditions is even more difficult. Therefore, one needs to account for set approximation techniques to find alternative conditions that are more conservative but easier to check. In the following, we consider spherical approximation, in which a set is overestimated by a bounding sphere. In this manner, approximate bisimulation relation is interpreted as error between the states and error between the inputs of the two systems. First, we restrict our attention to finding approximate bisimulation that is expressed in the following specific form:

$$R_{t,s} = \{x \mid \|x - \xi_s\| \leq \delta_{t,s}\}, \quad R_{t,a} = \{u \mid \|u - \xi_a\| \leq \delta_{t,a}\}. \quad (25)$$

Here, $\|\cdot\|$ denotes a norm, and $\delta_{t,s}$ and $\delta_{t,a}$ take positive values. The variable $\delta_{t,s}$ determines how much the state x of Σ can deviate from the state ξ_s of $\hat{\Sigma}$ at time t . Similarly, $\delta_{t,a}$ determines how much u may differ from ξ_a at time t . Given a norm, the set $R_{t,s}$ ($R_{t,a}$) is parametrized by a single variable $\delta_{t,s}$ ($\delta_{t,a}$).

Moreover, let us define the matrix-valued functions

$$\begin{aligned} A(x, u) &= \left[\frac{\partial f}{\partial x} + \frac{\partial f}{\partial u} \frac{\partial \phi}{\partial x} \right] (x, u), \\ B(x, u) &= \left[\frac{\partial f}{\partial u} \right] (x, u), \quad F(x) = \left[\frac{\partial \phi}{\partial x} \right] (x). \end{aligned} \quad (26)$$

Definition 6 A mesh $\mathcal{M}$ is feasible if there exists a set of variables $\{\delta_{t,s}\}, \{\delta_{t,a}\}$ ($t \in [0 : N], s \in \hat{\mathcal{S}}_t, a \in \mathcal{A}$) satisfying the following conditions:

$$1'') \text{ for all } s \in \mathcal{S}_0, \quad \delta_{0,s} \geq \bar{r}(C_s, \xi_s). \quad (27)$$

2'') for all (t, s, a, t', s') such that $(t, s) \xrightarrow{a} (t', s')$,

$$\|A\|_{t,s,a} \delta_{t,s} + \|B\|_{t,s,a} \bar{r}(C_a, \xi_a) + \|f(\xi_s, \xi_a) - \xi_{s'}\| \leq \delta_{t',s'} \quad (28a)$$

$$\|F\|_{t,s} \delta_{t,s} + \bar{r}(C_a, \xi_a) \leq \delta_{t,a} \quad (28b)$$

where

$$\begin{aligned} \|A\|_{t,s,a} &= \max_{\substack{\|x - \xi_s\| \leq \bar{r}(\bar{R}_{t,s}, \xi_s), \\ \|u - \xi_a\| \leq \bar{r}(\bar{R}_{t,a}, \xi_a)}} \|A(x, u)\|, \\ \|B\|_{t,s,a} &= \max_{\substack{\|x - \xi_s\| \leq \bar{r}(\bar{R}_{t,s}, \xi_s), \\ \|u - \xi_a\| \leq \bar{r}(\bar{R}_{t,a}, \xi_a)}} \|B(x, u)\|, \\ \|F\|_{t,s} &= \max_{\|x - \xi_s\| \leq \bar{r}(\bar{R}_{t,s}, \xi_s)} \|F(x)\|. \end{aligned} \quad (29)$$

3'') for all $t \in [0 : N]$, $s \in \hat{\mathcal{S}}_t$ and $a \in \mathcal{A}$,

$$\delta_{t,s} \leq \bar{r}(\bar{R}_{t,s}, \xi_s), \quad \delta_{t,a} \leq \bar{r}(\bar{R}_{t,a}, \xi_a). \quad (30)$$

Using these ingredients, we obtain the next theorem, which gives another sufficient condition for approximate bisimulation.

Theorem 2 Let Σ be a system, $\bar{R}$ be an error specification and ϕ be an input interface. Moreover, let $\mathcal{M}$ be a mesh satisfying Assumption 1. If $\mathcal{M}$ is feasible, then a set of binary relations R given by (25) is an approximate bisimulation between Σ and $\hat{\Sigma} = \text{QE}(\Sigma, \mathcal{M})$ that satisfies $\bar{R}$.

Proof: The conditions (27) and (30) directly imply (22) and (24), respectively. Next, for a tuple (t, s, a, t', s') such that $(t, s) \xrightarrow{a} (t', s')$, consider (x, u) such that $x \in R_{t,s}$ and $u \in C_a$. Here, $\|x - \xi_s\| \leq \delta_{t,s}$ and $\|u - \xi_a\| \leq \bar{r}(C_a, \xi_a)$ hold. From the mean value theorem, we have

$$f_s^\phi(x, u) = f(\xi_s, \xi_a) + \bar{A}(x - \xi_s) + \bar{B}(u - \xi_a)$$

where

$$\begin{aligned} \bar{A} &= \int_0^1 A(\xi_s + (x - \xi_s)\alpha, \xi_a + (u - \xi_a)\alpha) d\alpha, \\ \bar{B} &= \int_0^1 B(\xi_s + (x - \xi_s)\alpha, \xi_a + (u - \xi_a)\alpha) d\alpha. \end{aligned}$$

It also follows from (30) that $(x, \xi_s) \in \bar{R}_t^x$ and $(u, \xi_a) \in \bar{R}_t^u$ hold. Therefore, we obtain

$$\begin{aligned} \|f_s^\phi(x, u) - \xi_{s'}\| &= \|f(\xi_s, \xi_a) + \bar{A}(x - \xi_s) + \bar{B}(u - \xi_a) - \xi_{s'}\| \\ &\leq \|f(\xi_s, \xi_a) - \xi_{s'}\| + \|A\|_{t,s,a} \delta_{t,s} + \|B\|_{t,s,a} \bar{r}(C_a, \xi_a) \\ &\leq \delta_{t',s'}, \end{aligned}$$

which implies (23a). Here, the first inequality is from the triangle inequality and the second inequality is from (28a). From a similar discussion, we can show that (28b) implies (23b). Therefore, from Theorem 1, we conclude that R given by (25) is an approximate bisimulation satisfying $\bar{R}$. ■

In Theorem 2, equation (28a) describes how the accumulated error of one symbolic state is propagated to the other symbolic states. The term $\|A\|_{t,s,a} \delta_{t,s}$ expresses the error of s being propagated to its successor s' , the term $\|B\|_{t,s,a} \bar{r}(C_a, \xi_a)$

expresses the effect of input quantization error, and the term $\|f(\xi_s, \xi_a) - \xi_{s'}\|$ expresses the state quantization error. Therefore, Theorem 2 states that approximate bisimulation is achieved if there exists an error distribution over the states and inputs of the discrete model that satisfies the error propagation condition ((27), (28a) and (28b)) while not violating the error specification (30). In particular, an essential necessary condition is that there does not exist a cycle on the discrete abstract model whose “loop gain” (the product of $\|A\|_{t,s,a}$ s along the cycle) is greater than 1. This condition is indeed weaker than the global incremental stability. Consider a discrete model that has only two states, s_0 and s_1 , and no input. Suppose the transition relation is given as $s_0 \rightarrow s_1$, $s_1 \rightarrow s_0$. Moreover, suppose $\|A\|_{s_0} = 0.5$ and $\|A\|_{s_1} = 1.5$ (the subscript t is omitted for brevity). In this setting, the underlying system is obviously not globally incrementally stable since $\|A\|_{s_1}$ is greater than 1, but there exists an error distribution that satisfies (28a), for example, $\delta_{s_0} = 1.6$ and $\delta_{s_1} = 1.0$.

In (29), we overestimate the matrix norms considering the difference between (x, u) and (ξ_s, ξ_a) . For nonlinear systems, it is generally difficult to compute the max operations strictly. In such cases, the max operations are replaced with more conservative approximations. For example, when f and ϕ are C^1 , $\|A\|_{t,s,a}$ may be given by $\|A(\xi_s, \xi_a)\| + L(\underline{r}(\bar{R}_{t,s}, \xi_s) + \underline{r}(\bar{R}_{t,a}, \xi_a))$, where L denotes the Lipschitz constant of $\|A(x, u)\|$ (seen as a function of x and u).

V. FEASIBILITY CHECK AND DESIGN OF DISCRETE ABSTRACT MODELS

Based on the results obtained in the previous sections, in this section, we present practical methods for the feasibility check and design of discrete abstract models.

A. Feasibility Check of Discrete Abstraction

Before proceeding to the design problem, we discuss how to check whether a given mesh is feasible or not under a given error specification. A method for feasibility check is obtained almost directly based on Theorem 2 in Section IV. Notice that the conditions given in Theorem 2 can be expressed by a set of linear inequalities. In fact, the conditions (27)-(30) are rewritten in a single form given as

$$B\theta \leq Ax, x \leq \Theta. \quad (31)$$

Here, x denotes the vector of error variables, A and B are coefficient matrices, θ is a vector composed of cell radii and quantization errors and Θ is a vector that encapsulates the error specification. Concrete definition of these symbols are summarized in Table I (θ will appear later). Note that the components of θ can be made smaller by increasing the mesh resolution, while A and B , which are determined solely by the system dynamics, and Θ , which comes from the error specification, cannot be directly manipulated. As a result, the problem of finding an approximate bisimulation is formulated as a feasibility problem of linear inequality constraints. This problem, with an arbitrary objective function attached to it, can be solved efficiently by a linear programming (LP) solver. From Table I, we can see that the number of variables of this

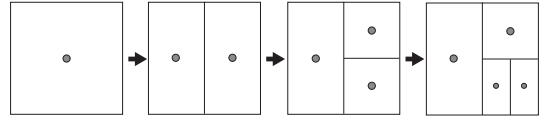


Fig. 2. An example of variable-resolution mesh

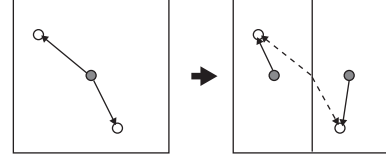


Fig. 3. Quantization errors before and after a subdivision.

LP is $O(N(|\mathcal{S}| + |\mathcal{A}|))$ using the big O notation, where $|\cdot|$ denotes the cardinality of a set. Here we overestimate $|\mathcal{S}_t|$ by $|\mathcal{S}|$. The number of inequality constraints is $O(N|\mathcal{S}||\mathcal{A}|)$.

B. Multi-resolution Quantizers

In this subsection, we characterize the notion of variable-resolution mesh. A variable-resolution mesh is a mesh with refinement operation. A refinement of a mesh is to partition a specified cell into two sub-cells. Moreover, it should be possible to make the quantization error at any specified point arbitrarily small by applying a proper sequence of refinements. In principle, any type of meshes satisfying this requirement is applicable to the design of discrete abstract models. In this paper, we consider a simple variable-resolution mesh, whose cells are orthogonal hyper-rectangles (Fig. 2). A mesh is initially composed of a single cell covering the entire domain. In a single refinement, a cell is subdivided into halves in the direction of one of the coordinate axes. For example, in the 2-dimensional case, a cell can be divided either horizontally or vertically. Each newly created cell is assigned a node placed at its center. This time, we assume that the selected cell is always subdivided in a direction in which it has the largest width. If it has the same width in all directions, it is subdivided in a random direction.

Remark 2 One may remove some of the above restrictions to obtain a feasible mesh with fewer cells. For example, the placement of nodes may be adjusted to reduce the quantization error $\|f(\xi_s, \xi_a) - \xi_{s'}\|$.

C. Iterative Mesh Refinement Algorithm

In Section V-A, we have seen that the feasibility of a mesh can be checked by solving a single linear program. Now, let us proceed one step further and consider how to design a feasible mesh. From a practical perspective, we would like to have a discrete abstract model not only approximately bisimilar to the concrete model but also composed of as small a number of symbolic states and symbolic inputs as possible; in other words, we would like the mesh as coarse as possible. One natural idea to achieve this goal is the following: initialize the mesh with a single cell covering the whole region and refine it

TABLE I
COMPONENT-WISE DESCRIPTION OF LINEAR INEQUALITIES (31)

range	$\mathbf{x}$	$\mathbf{A}$	$\mathbf{B}$	θ	$\tilde{\theta}$
$s \in \mathcal{S}_0$	$\delta_{0,s}$	1	1	$\bar{r}(C_s, \xi_s)$	$\bar{r}(C_s, \xi_s)$
$(t, s) \xrightarrow{a} (t', s')$	$\begin{bmatrix} \delta_{t',s'} \\ \delta_{t,s} \end{bmatrix}$	$\begin{bmatrix} 1 \\ -\ A\ _{t,s,a} \end{bmatrix}^T$	$\begin{bmatrix} \ B\ _{t,s,a} \\ 1 \end{bmatrix}^T$	$\begin{bmatrix} \bar{r}(C_a, \xi_a) \\ \ f(\xi_s, \xi_a) - \xi_{s'}\ \end{bmatrix}$	$\begin{bmatrix} \bar{r}(C_a, \xi_a) \\ \bar{r}(C_{s'}, \xi_{s'}) \end{bmatrix}$
$(t, s) \xrightarrow{a} (t', s')$	$\begin{bmatrix} \delta_{t,s} \\ \delta_{t,a} \end{bmatrix}$	$\begin{bmatrix} -\ F\ _{t,s} \\ 1 \end{bmatrix}^T$	1	$\bar{r}(C_a, \xi_a)$	$\bar{r}(C_a, \xi_a)$

(a) Components of $\mathbf{B}\theta \leq \mathbf{A}\mathbf{x}$

range	$\mathbf{x}$	Θ
$t \in [0 : N], s \in \hat{\mathcal{S}}_t$	$\delta_{t,s}$	$\underline{r}(\bar{R}_{t,s}, \xi_s)$
$t \in [0 : N], a \in \mathcal{A}$	$\delta_{t,a}$	$\underline{r}(\bar{R}_{t,a}, \xi_a)$

(b) Components of $\mathbf{x} \leq \Theta$

repeatedly until it becomes feasible. This refinement process can be generally captured as a search problem, in which one tries to find the shortest path (= refinement procedure) from the initial state (= a single cell) to a goal state (a feasible mesh). However, it is rather unrealistic to find the shortest refinement procedure, since the computational cost of the search problem can be prohibitively large even for very small systems. Instead, we take an iterative and greedy approach; starting from a state mesh and an input mesh both composed of a single cell, at each iteration, we make a refinement which seems to bring the mesh towards feasibility most quickly. This greedy method can be understood as a special type of search methods whose search depth is fixed to 1. In this case, we need some continuous measure of how far the current mesh is from feasibility. For this sake, let us add an extra variable to (31) to obtain the following linear program:

$$\begin{aligned} \text{LP}(\mathcal{M}) \quad & \text{minimize} \quad y \\ & \text{sub. to} \quad y \geq 0, \\ & \quad (1-y)\mathbf{B}\theta \leq \mathbf{A}\mathbf{x}, \mathbf{x} \leq \Theta. \end{aligned} \quad (32)$$

This LP has a new decision variable y , which controls the amount of relaxation of the constraint $\mathbf{B}\theta \leq \mathbf{A}\mathbf{x}$. Let us denote by $(\mathbf{x}^*, y^*)$ the optimal solution of $\text{LP}(\mathcal{M})$. Now, notice that $\text{LP}(\mathcal{M})$ is always feasible; it has a trivial solution $(\mathbf{x}, y) = (\mathbf{0}, 1)$. Moreover, if y^* is 0, $\mathbf{x}^*$ satisfies (31) and thus the mesh is feasible. From these facts, we can see that y^* ranges from 0 to 1, and it serves as a continuous measure of feasibility.

For later use, we also define the dual problem of $\text{LP}(\mathcal{M})$. Let λ and Λ be the dual variables corresponding to the inequality constraints $(1-y)\mathbf{B}\theta \leq \mathbf{A}\mathbf{x}$ and $\mathbf{x} \leq \Theta$, respectively. Then the dual problem is defined as

$$\begin{aligned} & \text{maximize} \quad (\mathbf{B}\theta)^T \lambda - \Theta^T \Lambda \\ & \text{sub. to} \quad \lambda \geq \mathbf{0}, \Lambda \geq \mathbf{0}, \\ & \quad \mathbf{A}^T \lambda - \Lambda = \mathbf{0}, (\mathbf{B}\theta)^T \lambda \leq 1. \end{aligned} \quad (33)$$

We denote the optimal solution of the dual problem by (λ^*, Λ^*) . When the primal problem is feasible, the dual

problem is also feasible and the optimal objective values of these two problems coincide; that is,

$$y^* = (\mathbf{B}\theta)^T \lambda^* - \Theta^T \Lambda^* \quad (34)$$

holds. Moreover, the complementarity slackness theorem states that $y^*(1 - (\mathbf{B}\theta)^T \lambda^*) = 0$. Since we consider the case where $y^* > 0$, we have $(\mathbf{B}\theta)^T \lambda^* = 1$ and thus $y^* = 1 - \Theta^T \Lambda^*$.

Our question is: which portion of the mesh should be refined, when $y^* > 0$? To answer this question, we first investigate the relation between y^* and θ because the components of θ reflect the mesh resolution. Here, we focus on the *active constraints* of LP, which are inequality constraints for which equalities hold at the optimal solution. The following lemma shows that the influence of θ on y^* can be measured based on the value of optimal dual variables.

Lemma 2 For $\text{LP}(\mathcal{M})$, consider a small change in θ such that the combination of the active constraints is unchanged. Here, the following relation holds:

$$\left(\frac{\partial y^*}{\partial \theta} \right)^T = (1 - y^*) \mathbf{B}^T \lambda^*. \quad (35)$$

Proof: By a proper reordering, we can write

$$\begin{aligned} (1 - y^*) \mathbf{B}_1 \theta &= \mathbf{A}_{11} \mathbf{x}_1^* + \mathbf{A}_{12} \mathbf{x}_2^*, & \lambda_1^* &> 0, \\ (1 - y^*) \mathbf{B}_2 \theta &< \mathbf{A}_{21} \mathbf{x}_1^* + \mathbf{A}_{22} \mathbf{x}_2^*, & \lambda_2^* &= 0, \\ \mathbf{x}_1^* &= \Theta_1, & \Lambda_1^* &> 0, \\ \mathbf{x}_2^* &< \Theta_2, & \Lambda_2^* &= 0 \end{aligned} \quad (36)$$

where

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{bmatrix}, \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}, \Theta = \begin{bmatrix} \Theta_1 \\ \Theta_2 \end{bmatrix}, \\ \mathbf{x}^* &= \begin{bmatrix} \mathbf{x}_1^* \\ \mathbf{x}_2^* \end{bmatrix}, \lambda^* = \begin{bmatrix} \lambda_1^* \\ \lambda_2^* \end{bmatrix}, \Lambda^* = \begin{bmatrix} \Lambda_1^* \\ \Lambda_2^* \end{bmatrix}. \end{aligned}$$

From $\mathbf{A}^T \lambda^* = \Lambda^*$ and (36), we have

$$\mathbf{A}_{11}^T \lambda_1^* = \Lambda_1^*, \mathbf{A}_{12}^T \lambda_1^* = 0. \quad (37)$$

Next, the partial differentiation of $y^* = 1 - \Theta_1^T \Lambda_1^*$ with respect to θ gives

$$\begin{aligned} \left(\frac{\partial y^*}{\partial \theta} \right)^T &= - \left(\frac{\partial \Lambda_1^*}{\partial \theta} \right)^T \Theta_1 = - \left(\frac{\partial \lambda_1^*}{\partial \theta} \right)^T A_{11} \Theta_1 \\ &= - \left(\frac{\partial \lambda_1^*}{\partial \theta} \right)^T ((1 - y^*) B_1 \theta - A_{12} x_2^*) \\ &= -(1 - y^*) \left(\frac{\partial \lambda_1^*}{\partial \theta} \right)^T B_1 \theta + \left(A_{12}^T \frac{\partial \lambda_1^*}{\partial \theta} \right)^T x_2^* \\ &= -(1 - y^*) \left(\frac{\partial \lambda_1^*}{\partial \theta} \right)^T B_1 \theta. \end{aligned} \quad (38)$$

The second and the fifth equality follows (37) and the third equality is from (36). Moreover, differentiating $(B_1 \theta)^T \lambda_1^* = 1$ yields

$$\left(\frac{\partial \lambda_1^*}{\partial \theta} \right)^T B_1 \theta + B_1^T \lambda_1^* = 0.$$

Substituting this into (38), we obtain

$$\left(\frac{\partial y^*}{\partial \theta} \right)^T = (1 - y^*) B_1^T \lambda_1^*,$$

which proves the lemma. ■

Using (35), we can estimate how much y^* will decrease after reducing a certain component of θ by looking at the corresponding component of $B^T \lambda^*$. Most LP solvers provide access to the optimal dual variables together with the optimal primal variables. Therefore, this way requires almost no extra computation.

Finally, we investigate how much each component of θ is reduced by a single subdivision of a cell. In Table I, we observe that θ consists of three different kinds of terms: $\bar{r}(C_s, \xi_s)$, $\bar{r}(C_a, \xi_a)$ and $\|f(\xi_s, \xi_a) - \xi_{s'}\|$. Each of the first two is simply the radius of a cell and therefore it can be reduced monotonically by subdividing the corresponding cell. The treatment of the third kind is slightly complicated. It would be natural to subdivide the cell $C_{s'}$ to reduce $\|f(\xi_s, \xi_a) - \xi_{s'}\|$. However, this term will not always be reduced after subdivision. As shown in Fig. 3, the quantization error may either decrease or increase by subdivision depending on the location of $f(\xi_s, \xi_a)$ in the cell $C_{s'}$. Here, we make use of the fact that $\|f(\xi_s, \xi_a) - \xi_{s'}\|$ is upper-bounded by $\bar{r}(C_{s'}, \xi_{s'})$, which can always be reduced by subdividing $C_{s'}$. Let $\hat{\theta}$ be a vector obtained by replacing all $\|f(\xi_s, \xi_a) - \xi_{s'}\|$ in θ with $\bar{r}(C_{s'}, \xi_{s'})$, as shown in Table I. Since every component of $\hat{\theta}$ is the radius of a cell, we can easily calculate which component of $\hat{\theta}$ will decrease by subdividing a certain cell, and how much it will decrease. Let $\Delta \hat{\theta}_{i,j}$ denote the reduction of the i -th component of $\hat{\theta}$ by the subdivision of the cell C_j . Here, C_j may be either a cell of the state mesh or a cell of the input mesh.

Based on the above discussion, we define the following heuristic rule for mesh refinement:

Mesh refinement heuristic: Subdivide the bottleneck cell C_{j^*} , where j^* is given by

$$j^* = \underset{j \in \mathcal{S} \cup \mathcal{A}, \bar{r}(C_j, \xi_j) \geq r_{\min}}{\operatorname{argmax}} \sum_i (B^T \lambda^*)_i \Delta \hat{\theta}_{i,j}. \quad (39)$$

TABLE II
ITERATIVE REFINEMENT ALGORITHM

Iterative Refinement Algorithm

```

initialize  $\mathcal{M}$ .
loop
  create quantizer embedding  $\hat{\Sigma} = \text{QE}(\Sigma, \mathcal{M})$ .
  solve  $\text{LP}(\mathcal{M})$ .  $y^* := \text{optimal cost}$ .
  if  $y^* = 0$ ,
    return  $\mathcal{M}$ .
  end
  if there is no bottleneck cell,
    return  $\emptyset$ .
  end
  refine  $\mathcal{M}$  at the bottleneck cell.
end

```

We call $\sum_i (B^T \lambda^*)_i \Delta \hat{\theta}_{i,j}$ the *influence* of the cell C_j . This term gives an estimate of the reduction of y^* by the subdivision of C_j . The bottleneck cell is a cell with the largest influence among all cells larger than $r_{\min}$. Here, the constant $r_{\min}$ is introduced to avoid subdividing excessively small cells.

The *Iterative Refinement Algorithm*, listed in Table II, constructs a feasible mesh by repeatedly subdividing the bottleneck cell. The algorithm is provided with a system Σ , an error specification $\bar{R}$ and a constant $r_{\min}$ specifying the minimum cell radius. First, each of the state mesh and the input mesh is initialized with a single cell. Here, the domain of each mesh is set to satisfy Assumption 1. At every iteration, a discrete model is constructed by quantizer-embedding using the current mesh. More precisely, at first, symbolic state transitions of the discrete model are computed. Next, for each transition $(t, s) \xrightarrow{a} (t', s')$, the coefficient matrices of a local linear system as well as their induced norms (29) are computed. To check the feasibility of the current mesh, a linear program $\text{LP}(\mathcal{M})$, defined in (32), is computed. Any solver can be used for solving $\text{LP}(\mathcal{M})$, as long as it provides access to the values of dual variables at the optimal solution. As discussed earlier, if the optimal cost y^* is 0, then the mesh is feasible and therefore it is outputted without further refinements. If $y^* > 0$, the mesh is refined at the bottleneck cell. By definition, the algorithm never subdivides a cell smaller than $r_{\min}$. Therefore, if all cells with positive influence are smaller than $r_{\min}$, the algorithm terminates the refinement process and outputs $\emptyset$, indicating that no feasible mesh was found. It is rather obvious that the algorithm will terminate in finite time; it will terminate, at the latest, when the resolution of $\mathcal{M}$ becomes smaller than $r_{\min}$. However, there remains a possibility that the algorithm might terminate with $\emptyset$ before finding a feasible mesh, even if one exists. The following theorem guarantees that the algorithm will output a feasible mesh if $r_{\min}$ is set small enough.

Theorem 3 *The Iterative Refinement Algorithm terminates in finite iterations. Moreover, if any mesh whose resolution is smaller than $r_{\min}$ is feasible, then the algorithm outputs a feasible mesh.*

Proof: By definition, the algorithm never subdivides a cell smaller than $r_{\min}$. Moreover, the domains of the state mesh and the input mesh, $\text{dom}(M^x)$ and $\text{dom}(M^u)$, are both bounded. Therefore, the first statement obviously holds. Now, suppose we have an infeasible mesh $\mathcal{M}$ at some point of iteration. Consider a submesh of $\mathcal{M}$, $\hat{\mathcal{M}}$, composed of all cells with positive influence. Here, any mesh that contains $\hat{\mathcal{M}}$ as a submesh is infeasible. From the assumption, on the other hand, any mesh whose resolution is smaller than $r_{\min}$ is feasible. This implies that there exists at least one cell in $\hat{\mathcal{M}}$ that is larger than $r_{\min}$, because otherwise one can construct a mesh with the resolution smaller than $r_{\min}$ that contains $\hat{\mathcal{M}}$ as a submesh, which is thus infeasible and contradicts the assumption. Therefore, the algorithm will not terminate with $\emptyset$ as its output. This proves the second statement and thus concludes the proof. ■

Theorem 3 states that if one knows a constant r such that any mesh whose resolution is below r is feasible, then the algorithm set as $r_{\min} = r$ will output a feasible mesh in finite time. In general, however, one does not know a priori whether such a constant exists, and how much such a constant will be. When the system is known to be globally incrementally stable, one can determine $r_{\min}$ based on the existing results of approximately bisimilar symbolic abstraction (for example, [11]). In this case, the algorithm will produce a mesh coarser than a uniform mesh with the resolution $r_{\min}$. As previously mentioned, a feasible mesh may exist even if the system is not globally incrementally stable. One way to find an appropriate $r_{\min}$ in such a case is to adjust $r_{\min}$ in outer loop: Initially set a large $r_{\min}$. Run the algorithm. If it outputs a feasible mesh, then terminate. Otherwise, decrease $r_{\min}$ with a certain rate. Repeat this until the algorithm produces a feasible mesh. Finally, the algorithm also provides information about the existence of a feasible mesh; if it outputs $\emptyset$ with some $r_{\min}$, then one can conclude that there exists at least one infeasible mesh with the resolution smaller than $r_{\min}$.

D. Notes on Implementation

It is the computation of linear programs that takes the major portion of the entire computation time. Therefore, from a practical point of view, keeping linear programs to be solved in small size is of crucial importance. In the following, we show that the size of the linear program $\text{LP}(\mathcal{M})$ in (32) can be decomposed into smaller and mutually independent subproblems by making use of the structure of the discrete abstract model. The largest of those sub-LPs has the size equal to the number of symbolic states and inputs that form cycles in the symbolic transitions. First, we separate the set of variables $\{\delta_{t,s}\}$ into *cyclic variables* and the others (non-cyclic variables). A variable $\delta_{t,s}$ is a cyclic variable if $t = N$ and s is a part of a cyclic path of the discrete abstract model. Here, let us denote by $\mathcal{S}_{N+}^c$ a subset of $\mathcal{S}_{N+}$ such that $\delta_{N,s}$ ($s \in \mathcal{S}_{N+}^c$) are cyclic variables.

Next, we define new lower-bound constraints for non-cyclic variables. For non-cyclic variables with $t = 0$, define $\theta_{0,s} =$

$\bar{r}(C_s, \xi_s)$, and for the other non-cyclic variables, define

$$\theta_{t',s'} = \max_{(t,s) \xrightarrow{a} (t',s')} [\|A\|_{s,a} \theta_{t,s} + \|B\|_{s,a} \bar{r}(C_a, \xi_a) + \|f(\xi_s, \xi_a) - \xi_{s'}\|]$$

recursively in the predecessor-to-successor order. Then, the original LP is decomposed into the following subproblems. One is an LP for the set of cyclic variables:

find $\{\delta_{N,s} \mid s \in \mathcal{S}_{N+}^c\}$ that minimizes y subject to
i) for all $s \in \mathcal{S}_{N+}^c$,

$$\begin{aligned} \delta_{N,s} &\leq \underline{r}(\bar{R}_{N,s}, \xi_s), \\ \delta_{N,s} &\geq (1-y)\bar{r}(C_s, \xi_s) \quad \text{if } N=0 \text{ and } s \in \mathcal{S}_0, \end{aligned}$$

ii) for all (s, a, s') such that $s \in \mathcal{S}_{N+}^c$, $s' \in \mathcal{S}_{N+}^c$ and $s \xrightarrow{a} s'$,

$$\begin{aligned} \delta_{N,s'} - \|A\|_{s,a} \delta_{N,s} &\geq \\ (1-y)\{ \|B\|_{s,a} \bar{r}(C_a, \xi_a) + \|f(\xi_s, \xi_a) - \xi_{s'}\| \}, \\ \delta_{N,a} - \|F\|_s \delta_{N,s} &\geq (1-y)\bar{r}(C_a, \xi_a), \\ \delta_{N,a} &\leq \underline{r}(\bar{R}_{N,a}, \xi_a), \end{aligned}$$

and iii) for all (s, a, s') such that $s \in (\mathcal{S}_{N-1} \cup \mathcal{S}_{N+} \setminus \mathcal{S}_{N+}^c)$, $s' \in \mathcal{S}_{N+}^c$ and $s \xrightarrow{a} s'$,

$$\begin{aligned} \delta_{N,s'} &\geq \\ (1-y)\{ \|A\|_{s,a} \theta_{t,s} + \|B\|_{s,a} \bar{r}(C_a, \xi_a) + \|f(\xi_s, \xi_a) - \xi_{s'}\| \}. \end{aligned}$$

The other subproblems are small LPs, one for each non-cyclic variable $\delta_{t,s}$:

minimize y subject to

$$\begin{aligned} \delta_{t,s} &\geq (1-y)\theta_{t,s}, \\ \delta_{t,s} &\leq \underline{r}(\bar{R}_{t,s}, \xi_s) \end{aligned}$$

and for all a such that $(t, s) \xrightarrow{a} (t', s')$ for some (t', s') ,

$$\|F\|_s \delta_{t,s} + (1-y)\bar{r}(C_a, \xi_a) \leq \underline{r}(\bar{R}_{t,a}, \xi_a).$$

The optimal cost of the original LP, y^* , is obtained by computing the optimal values of y for all subproblems and taking the largest value among them. Moreover, the LP for each non-cyclic variable involves only one variable $\delta_{t,s}$ and therefore it can be solved analytically. Therefore, by using this decomposition, the number of decision variables in the LP that must be computed using an LP solver is reduced to the number of cyclic variables, which is generally much smaller than the size of the original LP. The most extreme case is when finite-step approximate bisimulation is considered. In this case, all variables are non-cyclic variables and therefore the whole LP can be solved analytically. Although not explained in detail, the dual variables can also be obtained based on the solution of the subproblems.

The second expensive computation is the reconstruction of the discrete abstract model after a cell is subdivided. Since the subdivision of a single cell makes only local changes in the transition relations, one can save the computation effort by modifying only the affected portion of the existing discrete abstract model, instead of rebuilding it entirely.

TABLE III
GROWTH OF # OF CELLS WITH RESPECT TO N

N	1	2	3	4	5	10	10 + (7b)
state cells	98	136	146	146	149	153	154
input cells	2	4	4	4	6	7	7

(a) Uniform error ($\delta_0 = 0.3, \delta_1 = 0.0$)

N	1	2	3	4	5	10	10 + (7b)
state cells	46	71	98	126	136	130	130
input cells	2	4	4	8	10	10	10

(b) Relative error ($\delta_0 = 0.2, \delta_1 = 0.5$)

E. Comparison with Bisimulation Function-based Approaches

Some of the previous works take an alternative formulation based on bisimulation functions [10]. A bisimulation function gives approximate bisimulation relations as its level-sets. Using a bisimulation function, one can determine the resolution of discrete abstraction as an explicit function of a required precision. Bisimulation functions have been utilized to construct a discrete abstract model for a stable switched system in [11], and for efficient model checking by merging neighboring states in [16] and [12]. However, in order to guarantee infinite-step approximate bisimulation, one needs a contractive bisimulation function, which is found only for systems with the so-called global incremental stability. Moreover, to date, the forms of bisimulation functions are limited to simple types, which result in a discrete model whose state set takes the form of a lattice. Unlike the bisimulation function-based approach, our method computes an approximate bisimulation relation once a discrete model and an error specification are both given. Thanks to the error propagation analysis, the proposed framework is able to handle broader class of systems and error specifications. For the design of a discrete abstract model, however, we need to take an iterative procedure. Nevertheless, this procedure is fully automated.

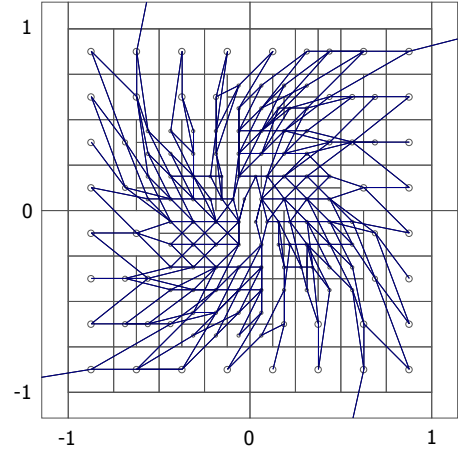
VI. EXAMPLES

This section shows some examples. The examples are chosen to highlight the properties of the proposed method from different perspectives. All computation is performed on a personal computer equipped with 1.0GHz CPU and 2.0GB memory. For an LP solver, we employed glpk [25].

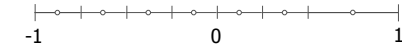
A. Linear System

In this example, we consider the discrete abstraction of a simple linear system and investigate how the complexity of the discrete model, measured by the number of cells, is affected by the number of time steps N and the required precision. Consider the following 2-dimensional linear system:

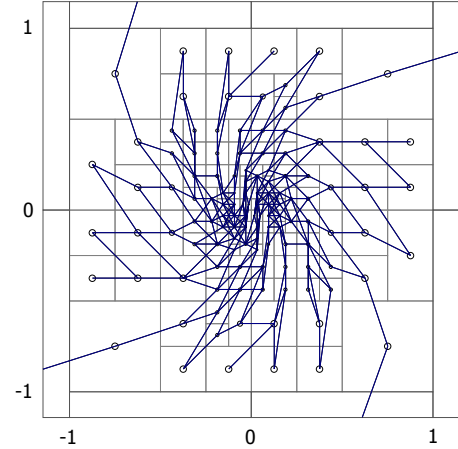
$$x_{t+1} = Ax_t + Bu_t, \quad A = \begin{bmatrix} 0.68 & -0.14 \\ 0.14 & 0.68 \end{bmatrix}, \quad B = \begin{bmatrix} 0.0 \\ 0.1 \end{bmatrix}.$$



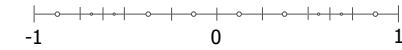
(a) State mesh ($\delta_0 = 0.3, \delta_1 = 0.0$)



(b) Input mesh ($\delta_0 = 0.3, \delta_1 = 0.0$)



(c) State mesh ($\delta_0 = 0.2, \delta_1 = 0.5$)



(d) Input mesh ($\delta_0 = 0.2, \delta_1 = 0.5$)

Fig. 4. Discrete abstract models of a linear system using different error specifications

The state set is given as $X = [-1, 1] \times [-1, 1]$ and the input set is given as $U = [-1, 1]$. The input interface is the common input type; $\phi(x, \hat{x}) = 0$. Therefore, the domain of the input mesh is simply set as $\text{dom}(M^u) = U$. The domain of the state mesh is set as $\text{dom}(M^x) = [-2.0, 2.0] \times [-2.0, 2.0]$ to satisfy the conditions in Assumption 1. For this system, we consider error specifications of the form $\bar{R}_t^x = \{(x, \hat{x}) \mid \|x - \hat{x}\| \leq \delta_0 + \delta_1 \|\hat{x}\|\} \quad (\forall t \in [0 : N])$, where $\delta_0 > 0$ and $\delta_1 \geq 0$ are the parameters. This type of error specifications with $\delta_1 > 0$ require high precision near the origin. By setting δ_1 as 0, we require constant precision in the entire state domain. No error specification is imposed on control inputs.

TABLE IV
GROWTH OF # CELLS WITH RESPECT TO PRECISION

δ_0	0.5	0.4	0.3	0.2	0.1
state cells	61	100	154	419	1704
input cells	4	4	7	8	16
(a) Uniform error ($\delta_1 = 0$)					
δ_0	0.5	0.4	0.3	0.2	0.1
state cells	40	60	84	130	422
input cells	2	4	6	10	22
(b) Relative error ($\delta_1 = 0.5$)					

Figs. 4(a)-(d) show discrete abstract models computed in to settings: $(\delta_0, \delta_1) = (0.3, 0.0)$ and $(\delta_0, \delta_1) = (0.2, 0.5)$. The parameter N is set as 10; we consider infinite-step approximate bisimulation with 10 steps of transient time interval. For the state meshes, only portions that correspond to the range $[-1.0, 1.0] \times [-1.0, 1.0]$ are displayed. Each line connecting nodes of the state mesh indicates that there exists a transition between the connected nodes with some input value. Slight asymmetry in partitioning is inevitable since the algorithm refines a mesh on one cell at a time. We can observe that in the relative error case, the state mesh has higher resolution near the origin, reflecting the error specification, while the total number of cells (130) is less than that in the uniform error case (154) (See Table III).

In the following, we investigate the growth of the number of cells with respect to various parameter changes. First, we consider the number of time steps N . We compare finite-step settings, in which the stationarity condition (7b) is not taken into account, varying N from 1 to 10. We also compare them with an infinite-step case, where (7b) is taken into account and N is set as 10. In Table III, we observe that in both of the uniform error and the relative error cases, the number of cells shows mild growth as N increases, and it shows almost no increase when the stationarity condition (7b) becomes effective. However, this may depend on the stability of the system considered. Next, we discuss the effect of the change in the precision parameter δ_0 . Here, we consider the infinite-step setting with $N = 10$. The precision parameter δ_0 is changed from 0.5 to 0.1 while δ_1 is fixed. As we observe in Table IV, the precision parameter δ_0 has more direct influence on the number of cells compared to the step length N . In the uniform error case shown in Table IV(a), the number of cells is approximately multiplied by four when the precision is doubled. This is because the state dimension of the system is two. In the relative error case shown in Table IV(b), on the other hand, the growth rate of the number of cells is less than exponential.

B. The van der Pol Oscillator

In this subsection, we describe the design flow of a discrete abstract model on nonlinear dynamical systems, taking the van

der Pol oscillator as an example. The van der Pol oscillator is originally governed by the following ordinary differential equation:

$$\ddot{x} = \mu(1 - x^2)\dot{x} - x. \quad (40)$$

Before applying our method, this system must be transformed into a discrete-time system. This time, we apply the Euler stepping to the first-order form of (40) to obtain

$$\begin{aligned} x_{t+1} &= f(x_t), \quad x_t = \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix}, \\ f(x) &= \begin{bmatrix} x_1 + (\mu(1 - x_2^2)x_1 - x_2)h \\ x_2 + x_1h \end{bmatrix}, \end{aligned} \quad (41)$$

which is a 2-dimensional discrete-time system. The parameters in the model are set as $\mu = 0.5, h = 0.15$, where h denotes the step size.

To compute a discrete abstract model of this system, at first we specify the target state set as $X = [-3, 3] \times [-3, 3]$. We consider finite-step approximate bisimulation with the number of time steps $N = 6$; the stationarity condition (7b) is not taken into account. The error specification is given as

$$\bar{R}_t^x = \{(x, \hat{x}) \mid \|x - \hat{x}\| \leq 0.5\} \quad (42)$$

for all $t \in [0 : N]$. The domain of the state mesh is set as $\text{dom}(M^x) = [-3.5, 3.5] \times [-3.5, 3.5]$, slightly larger than the target region X , in order to satisfy Assumption 1.

At each iteration of the mesh refinement algorithm, the coefficients of the linear program $\text{LP}(\mathcal{M})$ (32) must be calculated. The calculation of the coefficients $\|A\|_{t,s,a}$, $\|B\|_{t,s,a}$ and $\|F\|_{t,s}$, given by (29), is not straightforward for nonlinear systems because of the max operations. In this example, however, only $\|A\|_{t,s,a}$ needs to be calculated and it is dependent only on the symbolic state s (thus it can be written as $\|A\|_s$). This is because the system (41) has no input and the error specification is time-invariant. From (41), we have

$$A(x) = \frac{\partial f}{\partial x}(x) = \begin{bmatrix} 1 + \mu h(1 - x_2^2) & -h(1 + 2\mu x_1 x_2) \\ h & 1 \end{bmatrix}. \quad (43)$$

For a symbolic state s , $\|A\|_s$ is given by $\|A\|_s = \max_{\|x - \xi_s\| \leq 0.5} \|A(x)\|$. This time, we compute the max operation approximately by means of sampling (the detail is omitted). The components of θ are cell radii and quantization errors, which are not difficult to compute even for nonlinear system cases. All components of Θ are set as 0.5, reflecting the error specification (42). Since we consider finite-step bisimulation, the linear program $\text{LP}(\mathcal{M})$ can be computed efficiently using the decomposition technique described in Subsection V-D.

Fig. 5 shows a discrete abstract model of the van der Pol oscillator. In this result, the discrete abstract model consists of 3005 discrete states. We can observe that the obtained state mesh exhibits non-uniform resolution reflecting the system's nonlinear dynamics, even though the error specification is uniform. In particular, the mesh is finely partitioned in regions where the flow rapidly changes its direction.

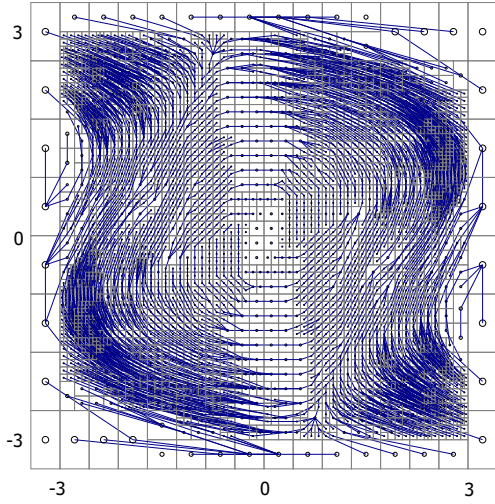


Fig. 5. Discrete abstract model of val der Pol oscillator.

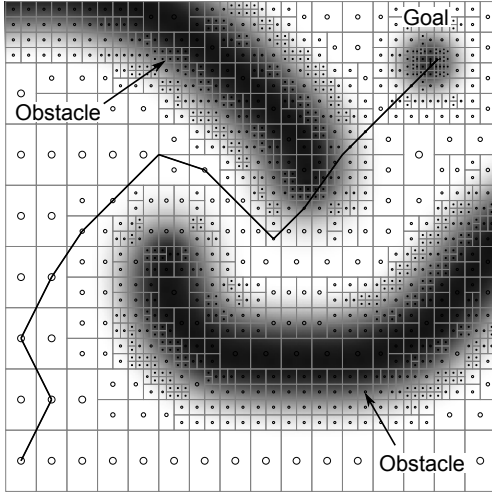


Fig. 6. Discrete abstract model of 2D workspace with obstacles

C. Finite-horizon Optimal Control

In this subsection, we describe with an example how discrete abstract models based on finite-step bisimulation can be used for solving finite-horizon optimal control problems, aiming at application to model predictive control. Before presenting an example, we describe a general procedure to obtain an approximate solution to a class of finite-horizon optimal control problems described below.

Problem 2 Finite Horizon Optimal Control

Let $\Sigma = \langle X, X_0, U, f \rangle$ be a system, $x_0 \in X_0$ be an initial state and N be a positive integer. Find a sequence of states and control inputs $(\{x_t\}_{0:N}, \{u_t\}_{0:N-1})$ that minimizes the cost function

$$J(\{x_t\}_{0:N}, \{u_t\}_{0:N-1}) = \sum_{t=0}^N [\psi_t^x(x_t) + \psi_t^u(u_t)] + \psi_N^x(x_N) \quad (44)$$

subject to $x_{t+1} = f(x_t, u_t)$ ($t \in [0 : N - 1]$).

In this setting, we do not consider rigid constraints; constraints are expressed by means of penalty functions. Let $\mathcal{P}$ be a map that returns an optimal trajectory $(\{x_t^*\}_{0:N}, \{u_t^*\}_{0:N-1})$ of Problem 2 for a given pair of a system Σ and an initial state x_0 ; we write $(\{x_t^*\}_{0:N}, \{u_t^*\}_{0:N-1}) = \mathcal{P}(\Sigma, x_0)$. Instead of computing $\mathcal{P}(\Sigma, x_0)$ directly, we construct an approximate solution of this problem based on $\mathcal{P}(\hat{\Sigma}, \hat{x}_0)$, where $\hat{\Sigma}$ is a discrete abstract model obtained by quantizer embedding, $\hat{\Sigma} = \text{QE}(\Sigma, \mathcal{M})$, and $\hat{x}_0 = Q^x(x_0)$. Since $\hat{\Sigma}$ has only finite states and input values, $\mathcal{P}(\hat{\Sigma}, \hat{x}_0)$ can be computed by means of dynamic programming, whose computational complexity is $\mathcal{O}(N)$. Let us write $(\{\hat{x}_t^*\}_{0:N}, \{\hat{u}_t^*\}_{0:N-1}) = \mathcal{P}(\hat{\Sigma}, \hat{x}_0)$. This trajectory is then mapped to a trajectory $(\{x'_t\}_{0:N}, \{u'_t\}_{0:N-1})$ of Σ as follows:

$$\begin{aligned} x'_0 &= x_0, \\ u'_t &= \hat{u}_t^* + \phi(x'_t, \hat{x}_t^*), \quad x'_{t+1} = f(x'_t, u'_t) \quad (t \in [0 : N - 1]) \end{aligned} \quad (45)$$

where the interface is defined in Section IV. Thus the trajectory $(\{x'_t\}_{0:N}, \{u'_t\}_{0:N-1})$ is regarded as an approximate solution of $\mathcal{P}$.

Now, let us discuss the relation between error specifications and sub-optimality. Consider the following error specification.

$$\begin{aligned} (x, \hat{x}) \in R_t^x &\Leftrightarrow \|\psi_t^x(x) - \psi_t^x(\hat{x})\| \leq \epsilon, \\ (u, \hat{u}) \in R_t^u &\Leftrightarrow \|\psi_t^u(u) - \psi_t^u(\hat{u})\| \leq \epsilon \end{aligned} \quad (46)$$

where ϵ is a positive constant. This error specification requires high precision in regions where the cost functions change rapidly. Then, we obtain the following result.

Theorem 4 Let Σ be a system and $\hat{\Sigma} = \text{QE}(\Sigma, \mathcal{M})$ be a discrete abstract model based on an error specification given by (46). Moreover, let $(\{x_t^*\}_{0:N}, \{u_t^*\}_{0:N-1}) = \mathcal{P}(\Sigma, x_0)$ with an initial state $x_0 \in X_0$, and let $(\{\hat{x}_t^*\}_{0:N}, \{\hat{u}_t^*\}_{0:N-1}) = \mathcal{P}(\hat{\Sigma}, \hat{x}_0)$ where $\hat{x}_0 = Q^x(x_0)$. Furthermore, let $(\{x'_t\}_{0:N}, \{u'_t\}_{0:N-1})$ be a trajectory given by (45). Then, the following relation holds:

$$\begin{aligned} J(\{x'_t\}_{0:N}, \{u'_t\}_{0:N-1}) \\ \leq J(\{x_t^*\}_{0:N}, \{u_t^*\}_{0:N-1}) + 2(2N + 1)\epsilon. \end{aligned} \quad (47)$$

Proof: By the definition of approximate bisimulation, there exists a trajectory of $\hat{\Sigma}$,

$(\{\hat{x}'_t\}_{0:N}, \{\hat{u}'_t\}_{0:N-1})$, satisfying $\hat{x}'_0 = \hat{x}_0$ and $(x_t^*, \hat{x}'_t) \in R_t^x$ ($t \in [0 : N]$), $(u_t^*, \hat{u}'_t) \in R_t^u$ ($t \in [0 : N - 1]$). From (46), we have

$$\begin{aligned} J(\{\hat{x}'_t\}_{0:N}, \{\hat{u}'_t\}_{0:N-1}) \\ \leq J(\{x_t^*\}_{0:N}, \{u_t^*\}_{0:N-1}) + (2N + 1)\epsilon. \end{aligned} \quad (48)$$

On the other hand, from a similar discussion, we have

$$\begin{aligned} J(\{x'_t\}_{0:N}, \{u'_t\}_{0:N-1}) \\ \leq J(\{\hat{x}'_t\}_{0:N}, \{\hat{u}'_t\}_{0:N-1}) + (2N + 1)\epsilon. \end{aligned} \quad (49)$$

Finally, the optimality of $(\{\hat{x}_t^*\}_{0:N}, \{\hat{u}_t^*\}_{0:N-1})$ implies

$$J(\{\hat{x}_t^*\}_{0:N}, \{\hat{u}_t^*\}_{0:N-1}) \leq J(\{\hat{x}'_t\}_{0:N}, \{\hat{u}'_t\}_{0:N-1}). \quad (50)$$

From (48), (49) and (50), we obtain (47). ■

This theorem claims that the cost of the approximate solution is upper-bounded by the sum of the optimal cost and a known constant.

In the following, we demonstrate the above result in a path-planning problem of a 2D vehicle. Consider an omnidirectional vehicle on a 2D plane. The state of the vehicle is simply the position in the workspace in x - y coordinate; $x_t = [p^x, p^y]^T$. The control input is given as the change in the vehicle's position; thus the state equation is given by $x_{t+1} = x_t + u_t$. The purpose of the vehicle is to reach the goal position from an initial position with the minimum cost. The cost function is defined as the sum of costs at the vehicle's position at each time instant; $\sum_{t=0}^N \psi^x(x_t)$. Here, the cost at each position is given by the sum of a penalty (positive cost) for moving near the obstacles and a reward (negative cost) for being close to the goal. In Fig. 6, the penalty and the reward are depicted in gray scale. The darkest regions of the obstacles are assigned the largest positive cost of 1.0, while the darkest region of the goal is assigned the largest negative cost of -1.0 . We consider finite-step bisimulation with N set as 15. We use an input interface of the form $\phi(x, \hat{x}) = \alpha(\hat{x} - x)$, where α is set as 0.8. For this input interface, the term $\|A\|_{t,s,a}$, which appears in (28a), is obtained as 0.2. This enables the design of discrete models with much lower resolution, compared to the common-input setting $\phi(x, \hat{x}) = 0$. The error specification is given by (46) with $\epsilon = 0.5$, except that in this example, no error specification is imposed on control inputs. Here, the error tolerances $\underline{r}(\bar{R}_{t,s}, \xi_s)$ under the specification given by (46) are computed approximately by sampling. We have obtained a state mesh composed of 1004 cells and an input mesh composed of 256 cells. The obtained state mesh is overlaid on the workspace shown in Fig. 6. We can observe that the mesh is finely partitioned near obstacle boundaries and around the goal, where an error in the state results in a significant change in the cost. In the figure, one optimal trajectory starting from an initial position located at the lower-left of the workspace is shown. Note that once a discrete abstract model is obtained, one can use it to compute an approximately optimal trajectory for an arbitrary initial position, as long as the obstacle configuration and the goal position is unchanged.

VII. CONCLUSION

In this paper, we have presented a new computational method for the feasibility check and design of discrete abstract models for nonlinear systems. First, we have proposed a new analytical scheme for approximate bisimulation based on the notion of error distribution and derived a condition for approximate bisimulation, which can be checked using a linear programming solver. Next, the Iterative Refinement Algorithm, which refines a variable-resolution mesh for quantizer embedding based on the information of active constraints provided by the LP solver, has been presented. The proposed algorithm is guaranteed to terminate in finite iterations and to produce a mesh that gives a feasible discrete abstract model, if the termination condition is set appropriately.

Thanks to non-uniform discretization, the present results enables us to design discrete abstract models composed of

a smaller number of states and inputs, compared to those obtained by uniform discretization. However, it is still challenging to apply this technique to practical control applications, in which the state space and input space may easily have more than ten dimensions. An effective way to cope with such large scale systems might be to decompose a system into an interconnection of smaller subsystems and apply the proposed technique to subsystems for which discretization is particularly useful (see [23] for further details).

Another future direction is to extend the proposed framework to the domain of continuous-time systems. By extending the quantizer embedding in such a way that not only states and control inputs but also time is quantized by means of a mesh structure, discrete abstraction of continuous-time systems will be put into the proposed framework. Moreover, resultant discrete abstract models may exhibit non-uniform discretization of time axis, reflecting the error specification and the system's dynamics.

REFERENCES

- [1] G. Lafferriere, G.J. Pappas and S. Sastry : Hybrid Systems with Finite Bisimulations, P. Antsaklis et al. (Eds.) : Hybrid Systems V, LNCS 1567, 186/203, 1999.
- [2] M.E. Broucke : A Geometric Approach to Bisimulation and Verification of Hybrid Systems, Hybrid Systems: Computation and Control (HSCC99), Lecture Notes in Computer Science 1569, Springer-Verlag, 61/75, 1999.
- [3] J. Lunze, B. Nixdorf and J. Schröder : Deterministic Discrete-Event Representations of Linear Continuous-Variable Systems, Automatica, Vol. 35, 395/406, 1999.
- [4] R. Alur, T.D. Verimag and F. Ivančić : Predicate Abstractions for Reachability Analysis of Hybrid Systems, ACM Trans. on Embedded Computing Systems, Vol. 5, No. 1, 152/199, 2006.
- [5] J. Lunze : A Timed Discrete-Event Abstraction of Continuous-Variable Systems, International Journal of Control, Vol. 72, No. 13, 1147/1164, 1999.
- [6] J. Raisch and S.D. O'Young : Discrete Approximation and Supervisory Control of Continuous Systems, IEEE Trans. on Automatic Control, Vol. 43, No. 4, 569/573, 1998.
- [7] H.L. Hagenars, J. Imura and H. Nijmeijer : Approximate Continuous-Time Optimal Control in Obstacle Avoidance by Time/Space Discretization of Non-convex State Constraints, Proc. of IEEE Conf. on Control Applications, 878/883, 2004.
- [8] J. Imura and H. Matsushima : Simultaneous Optimization of Continuous Control Inputs and Discrete State Waypoints, 9th International Workshop on Hybrid Systems: Computation and Control (HSCC 2006), J. Hespanha and A. Tiwari (Eds.) LNCS 3927 Springer-Verlag, 302/317, 2006.
- [9] R. Milner : Communication and Concurrency, Prentice-Hall, 1989.
- [10] A. Girard and G.J. Pappas : Approximation Metrics for Discrete and Continuous Systems, IEEE Transactions on Automatic Control, Vol. 52, No. 5, 782/798, 2007.
- [11] A. Girard, G. Pola and P. Tabuada : Approximately Bisimilar Symbolic Models for Incrementally Stable Switched Systems, IEEE Transactions of Automatic Control, Vol. 55, No. 1, 116/126, 2010.
- [12] G. Zheng and A. Girard : Bounded and Unbounded Safety Verification Using Bisimulation Metrics, 12th International Conference on Hybrid Systems: Computation and Control (HSCC2009), M. Rupak and P. Tabuada (Eds.) LNCS 5469 Springer-Verlag, 426/440, 2009.
- [13] A. Girard and G.J. Pappas : Hierarchical Control System Design Using Approximate Simulation, Automatica, Vol. 45, No. 2, 566/571, 2009.
- [14] P. Tabuada : Approximate Simulation Relations and Finite Abstractions of Quantized Control Systems, Hybrid Systems: Computation and Control, Vol. 4416 in LNCS, 529/542, Springer, 2007.
- [15] G. Pola, A. Girard and P. Tabuada : Symbolic Models for Nonlinear Control Systems Using Approximate Bisimulation, 46th IEEE Conference on Decision and Control, 4656/4661, 2007.
- [16] F. Lerda, J. Kapinski, M. Clarke and B.H. Krogh : Verification of Supervisory Control Software Using State Proximity and Merging, Hybrid Systems: Computation and Control, Vol. 4981 in LNCS, 344/357, Springer, 2008.

- [17] X.D. Koutsoukos, P.J. Antsaklis, J.A. Stiver and M.D. Lemmon : Supervisory Control of Hybrid Systems, In Proc. of IEEE, Special Issue in Hybrid Systems. P.J. Antsaklis, Ed., 1026/1049, July 2000.
- [18] L.C.G.J.M. Habets, P.J. Collins and J.H. van Schuppen : Reachability and Control Synthesis for Piecewise-Affine Hybrid Systems on Simplices, IEEE Transactions on Automatic Control, Vol. 51, No. 6, 938/948, 2006.
- [19] M. Kloetzer, C. Belta : A Fully Automated Framework for Control of Linear Systems from LTL Specifications, Hybrid Systems: Computation and Control (HSCC06), Lecture Notes in Computer Science 3927, Springer-Verlag, 333/347, 2006.
- [20] B. Yordanov and C. Belta : Temporal Logic Control of Discrete-Time Piecewise Affine Systems, Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference, 3182/3187, Shanghai, P.R. China, December 15-18, 2009.
- [21] R. Munos and A. Moore : Variable Resolution Discretization in Optimal Control, Machine Learning Journal, Vol. 49, No. 2-3, 291/323, 2001.
- [22] Y. Tazaki, J. Imura; Finite Abstractions of Discrete-time Linear Systems and Its Application to Optimal Control, 17th IFAC World Congress, 2008.
- [23] Y. Tazaki, J. Imura; Bisimilar Finite Abstractions of Interconnected Systems, Hybrid Systems: Computation and Control, Vol. 4981 in LNCS, 514/527, Springer, 2008.
- [24] Y. Tazaki and J. Imura : Discrete-State Abstractions of Nonlinear Systems Using Multi-resolution Quantizer, Hybrid Systems: Computation and Control, Vol. 5469 in LNCS, 351/365, Springer, 2009.
- [25] GLPK (GNU Linear Programming Kit), <http://www.gnu.org/software/glpk/>.

PLACE
PHOTO
HERE

Yuichi Tazaki Yuichi Tazaki was born in Kanagawa, Japan, in 1980. He received the Dr.Eng degree in control engineering from Tokyo Institute of Technology, Tokyo, Japan, in 2008. From 2007 to 2009, he was a research fellow of Japan Society for the Promotion of Science. In 2008, he was a guest scientist of Honda Research Institute Europe, Germany. From 2009, he has been an assistant professor of Nagoya University, Japan. His research interests include abstraction and control of hybrid systems, control of bipedal robots and motion planning of

humanoid robots. He is a member of IEEE, SICE and The Robotics Society of Japan.

PLACE
PHOTO
HERE

Jun-ichi Imura Jun-ichi Imura was born in Gifu, Japan, in 1964. He received a M.S. degree in Applied Systems Science, and a Ph.D. degree in Mechanical Engineering from Kyoto University, Japan, in 1990 and 1995, respectively. He served as a Research Associate at the Department of Mechanical Engineering, Kyoto University from 1992 to 1996, and as an Associate Professor at the Division of Machine Design Engineering, Faculty of Engineering, Hiroshima University from 1996 to 2001. From May 1998 to April 1999 he was a visiting researcher at

the Faculty of Mathematical Sciences, University of Twente, The Netherlands. Since 2001 he has been with the Department of Mechanical and Environmental Informatics, Graduate School of Information Science and Engineering, Tokyo Institute of Technology, where he is currently a Professor. His research interests include control of nonlinear systems, hybrid systems, and large-scale systems. He is an Associate Editor of Automatica, and an Associate Editor of SICE Journal of Control, Measurement, and System Integration. He is a member of IEEE, SICE, ISCIE, IEICE, and The Robotics Society of Japan.