Approximately Bisimilar Discrete Abstractions of Nonlinear Systems Using Variable-resolution Quantizers

Yuichi Tazaki and Jun-ichi Imura

***Abstract*— This paper presents a method for the design of discrete abstract models of nonlinear continuous-state systems under the framework of approximate bisimulation. First, the notion of quantizer embedding, which transforms a continuous-state system into a finite-state system, is extended to a variable-resolution setting. Next, it is shown that the series of conditions for approximate bisimulation can be converted into a set of linear inequalities, which can be verified by a linear programming solver. From this result, we obtain an algorithm that repeatedly refines a variable-resolution mesh until approximate bisimulation with a prescribed error specification is achieved.**

I. Introduction

This paper proposes a method for the design of discrete abstractions of nonlinear systems using variable-resolution quantizers. Recently, discrete abstraction methods based on approximate bisimulation [1][2] has gained growing attention. Approximate bisimulation is an extension of the original bisimulation to metric space. It admits equivalence relation between two systems if the distance of output signals can be kept within a given threshold. Until now, it has been shown that discrete abstractions of a wide range of systems can be obtained based on approximate bisimulation (see [3][4][5]). Discrete abstraction based on approximate bisimulation is especially suitable for control problems with a quantitative performance measure. It also has an advantage that it does not require expensive geometric computations for constructing a discrete abstract model. To date, however, it has the following limitations. First, the error specification in the conventional approximate bisimulation has to be uniform. From a practical perspective, it is desirable to support non-uniform error specification (for an example, error margin proportional to the norm of the signal itself). Second, the distribution of discrete states is also uniform. This means that the number of discrete states grows exponentially with respect to the dimension of the state space. As the third point, it requires the global incremental stability [5]. Since many nonlinear systems may have local incrementally unstable regions, this requirement may severely limit the range of application. The use of variable-resolution quantizers enables the design of discrete abstract models with non-uniformly distributed states and control inputs.

In [7], the authors have presented a computational method for the design of discrete abstract models that iteratively refines a variable-resolution mesh until a given error specification is met. However, the method has limitations that it can guarantee error specifications only in finite time steps and that it assumes common input signals are applied to two systems.

The method presented in this paper is an extension to our previous method, which is capable of design discrete abstract models under conditions much closer to approximate bisimulation; that is, it can handle infinite-step error specification, and moreover, the control input signals of the two systems may be different. In addition, it provides us with a new view of discrete abstraction problems based on mathematical programming.

Notation: For a function f and a set C, we write $f(C) = \{f(x) \,|\, x \in C\}$. Similar notation is used for multivariate functions. The symbol $\mathbb{R}$ denotes the field of real numbers and the symbol $\mathbb{Z}^+$ denotes the set of non-negative integers.

II. Approximate Bisimulation of Discrete-time Systems

In this paper, we address the discrete abstraction of discrete-time continuous-state systems. A discrete-time system is a tuple $\langle X, X_0, U, f\rangle$, where $X \subset \mathbb{R}^n$ is the set of states, $X_0 \subseteq X$ is the set of initial states, $U \subset \mathbb{R}^m$ is the set of control inputs and $f : X \times U \mapsto X$ is the state transition function. The state and the control input of the system at time $t \in \mathbb{Z}^+$ are expressed as $x_t \in X$ and $u_t \in U$, respectively. The state transition at time t is expressed as

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

We assume that the sets X and U are both bounded and the function f is smooth. Moreover, we assume that any trajectory of Σ starting from $x_0 \in X_0$ will never exceed X.

In the following, we introduce the notion of approximate bisimulation for the class of systems defined above. Approximate bisimulation ([1]) is a framework for comparing a symbolic system with a continuous-state system in a quantitative sense. It admits an equivalence relation between two systems if each system can *simulate* the other while maintaining a given error criterion imposed on their signals.

Definition 1 *Approximate 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 and let $\bar{R}^{\mathrm{x}} \subset \mathbb{R}^n \times \mathbb{R}^n$ be a binary relation of states and $\bar{R}^{\mathrm{u}} \subset \mathbb{R}^m \times \mathbb{R}^m$ be a binary relation of control inputs. A binary relation $R^{\mathrm{x}} \subset \mathbb{R}^n \times \mathbb{R}^n$ is an approximate bisimulation relation with respect to $(\bar{R}^{\mathrm{x}}, \bar{R}^{\mathrm{u}})$

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

if and only if the following conditions hold:
1) for any $x \in X_0$, there exists a $\hat{x} \in \hat{X}_0$ such that

$$(x, \hat{x}) \in R^{\mathrm{x}} \tag{2}$$

and for any $\hat{x} \in \hat{X}_0$, there exists a $x \in X_0$ such that (2) holds.
2) for any $(x, \hat{x}) \in R^{\mathrm{x}}$, the following holds: for any $u \in U$, there exists a $\hat{u}$ such that

$$(u, \hat{u}) \in \bar{R}^{\mathrm{u}},\ (f(x,u), \hat{f}(\hat{x}, \hat{u})) \in R^{\mathrm{x}} \tag{3}$$

holds, and for any $\hat{u} \in \hat{U}$, there exists a u such that (3) holds.
3) the following holds:

$$R^{\mathrm{x}} \subseteq \bar{R}^{\mathrm{x}}. \tag{4}$$

In Definition 1, 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^{x}. Condition 2 requires that if a state pair is in the relation R^{x}, then for any control input of one system, there should exist a control input of the other system that satisfies the relation $\bar{R}^{\mathrm{u}}$ and drives the state so that the relation R^{x} is satisfied at the next time instant. In other words, the two systems should be able to *simulate* each other. The relation $\bar{R}^{\mathrm{u}}$ encodes an *error specification* imposed on control input variables of the two systems. Note that the input selected first must be an element of the bounded input set, but the input selected later may be outside this set as long as it satisfies the relation $\bar{R}^{\mathrm{u}}$ with the input selected first. In this sense, the sets U and $\hat{U}$ should be understood as an input range considered in approximate bisimulation rather than as a set in which the inputs must be strictly constrained. Finally, Condition 3 requires that any state pair $(x, \hat{x})$ that satisfies the relation R^{x} should satisfy the relation $\bar{R}^{\mathrm{x}}$, which encodes an error specification imposed on the state variables.

A simple example of $\bar{R}^{\mathrm{x}}$ is a uniform error condition: $\bar{R}^{\mathrm{x}} = \{(x, \hat{x}) \,|\, \|x - \hat{x}\| \leq \epsilon\}$, where ϵ is a positive constant. Only this type of error condition has been considered in the conventional approximate bisimulation. On the other hand, $\bar{R}^{\mathrm{x}} = \{(x, \hat{x}) \,|\, \|x - \hat{x}\| \leq \eta \min\{\|x\|, \|\hat{x}\|\} + \epsilon\}$, where both ϵ and η are positive constants, expresses a relative error condition.

III. Quantizer Embedding

In this section, we introduce the notion of *quantizer embedding* ([7]). First of all, we introduce *mesh* structure defined in a bounded region in $\mathbb{R}^n$. A mesh M^{x} is a finite collection of pairs denoted by

$$M^{\mathrm{x}} = \{(\xi^{\mathrm{x}}_0, C^{\mathrm{x}}_0), (\xi^{\mathrm{x}}_1, C^{\mathrm{x}}_1), \ldots, (\xi^{\mathrm{x}}_{|\mathcal{S}|}, C^{\mathrm{x}}_{|\mathcal{S}|})\}. \tag{5}$$

Each $C^{\mathrm{x}}_s\,(s \in [1 : |\mathcal{S}|])$ is a compact subset of $\mathbb{R}^n$ and called a *cell* of the mesh. The cells are mutually disjoint; $C^{\mathrm{x}}_i \cap C^{\mathrm{x}}_j = \emptyset$ $(i \neq j)$. A region covered by the cells is called the *domain* of the mesh and denoted by $\mathrm{dom}(M^{\mathrm{x}}) = \bigcup_s C^{\mathrm{x}}_s$. Each $\xi_s \in C_s$ is called the *node* of the s-th cell. Moreover,

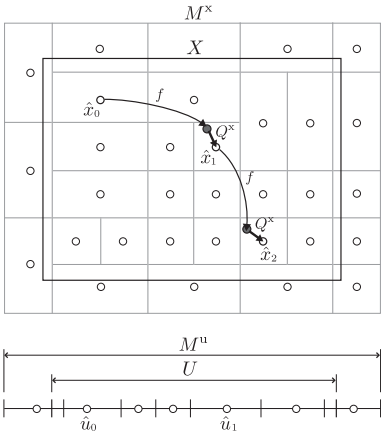


Fig. 1. State transition of a finite-state system obtained by quantizer-embedding.

we call a subset of M^{x} a *sub-mesh* of M^{x}. A mesh M^{x} defines a quantization function in $\mathrm{dom}(M^{\mathrm{x}})$ as shown below.

$$\begin{aligned} &Q[M^{\mathrm{x}}] \,:\, \mathrm{dom}(M^{\mathrm{x}}) \mapsto \{\xi^{\mathrm{x}}_0, \xi^{\mathrm{x}}_1, \ldots, \xi^{\mathrm{x}}_{|\mathcal{S}|}\}, \\ &Q[M^{\mathrm{x}}](x) = \xi^{\mathrm{x}}_s \ \text{ if } \ x \in C^{\mathrm{x}}_s. \end{aligned} \tag{6}$$

A quantization function $Q[M^{\mathrm{x}}](\cdot) =: Q^{\mathrm{x}}(\cdot)$ maps an arbitrary point $x \in \mathrm{dom}(M^{\mathrm{x}})$ to a node whose corresponding cell includes x. Let us call this function the state quantizer. Similarly, a mesh defined in the input space is denoted by

$$M^{\mathrm{u}} = \{(\xi^{\mathrm{u}}_0, C^{\mathrm{u}}_0), (\xi^{\mathrm{u}}_1, C^{\mathrm{u}}_1), \ldots, (\xi^{\mathrm{u}}_{|\mathcal{A}|}, C^{\mathrm{u}}_{|\mathcal{A}|})\}$$

and the input quantizer $Q[M^{\mathrm{u}}](\cdot) =: Q^{\mathrm{u}}(\cdot)$ is defined in the same way as the state quantizer. From now on, we call a pair of a state mesh and an input mesh, $\mathcal{M} = (M^{\mathrm{x}}, M^{\mathrm{u}})$, simply a mesh. A discrete-time system is transformed into a finite-state system by embedding a pair of quantizers into its state-transition function.

Definition 2 *Quantizer Embedding of Discrete-time Systems Let $\Sigma = \langle X, X_0, U, f\rangle$ be a discrete-time system. Moreover, let $\mathcal{M} = (M^{\mathrm{x}}, M^{\mathrm{u}})$ be a mesh. The quantizer embedding (QE in short) of Σ, denoted by $\mathrm{QE}(\Sigma, \mathcal{M})$, is a system*

$$\hat{\Sigma} = \langle Q^{\mathrm{x}}(X), Q^{\mathrm{x}}(X_0), Q^{\mathrm{u}}(U), \hat{f}\rangle$$

whose state transition function is defined as

$$\hat{f}(x, u) := Q^{\mathrm{x}}(f(x, u)). \tag{7}$$

As illustrated in Fig. 1, the control input of $\hat{\Sigma}$ is chosen from the nodes of M^{u}. Moreover, 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. Therefore, as long as the meshes M^{x} and M^{u} are composed of a finite number of cells, a system with a state-transition of the form (7) can be viewed as a finite automaton. In fact, the state transition of $\hat{\Sigma}$ can be rewritten in a symbolic form as

$$s \xrightarrow{a} s' \ \Leftrightarrow \ f(\xi^{\mathrm{x}}_s, \xi^{\mathrm{u}}_a) \in C^{\mathrm{x}}_{s'}. \tag{8}$$

From now on, we focus on a class of discrete abstract models that are obtained by means of the quantizer embedding of the original system. This reduces the discrete abstraction problem to the problem of designing a state mesh and an input mesh.

Although not discussed in detail, in order for the behavior of the quantizer-embedded system to be well-defined in the context of approximate bisimulation, it is required that the domain of the state mesh and that of the input mesh include X and U, respectively, with certain "margins".

IV. Derivation of Conditions for Approximate Bisimulation

In this section, we will transform a series of conditions in Definition 1 into a more tractable form that is well suited to computational methods. First, we observe that that existence quantifiers appear several times in the definition of approximate bisimulation, which are quite difficult to handle directly in the design of discrete abstraction. To transform them into tractable forms, we replace these quantifiers with explicit maps. We call these maps *interfaces*, borrowing the term from [6]. For Condition 1 in Definition 1, we assume 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 $\hat{x}_0 = Q^{\mathrm{x}}(x_0)$. In the opposite case, the initial state of Σ is set as $x_0 = \hat{x}_0$. Similarly, for Condition 2, the simulating input of one system is given by an explicit function of both states and the input of the other system; given x_t, $\hat{x}_t$ and u_t, $\hat{u}_t = Q^{\mathrm{u}}(u_t + \phi(\hat{x}_t, x_t))$, and in the opposite case, $u_t = \hat{u}_t + \phi(x_t, \hat{x}_t)$, where ϕ is a smooth function that satisfies $\phi(x, \hat{x}) = -\phi(\hat{x}, x)$. We call ϕ the *input interface*. In general, the use of an input interface enables discrete abstraction in coarser resolution, compared to the common control input setting ($\phi(x, \hat{x}) \equiv 0$).

The next step is to decompose the approximate bisimulation relation with respect to the states and inputs of the discrete abstract model. To this aim, we introduce the following symbols:

$$\begin{aligned} &R^{\mathrm{x}}_s := \{x \,|\, (x, \xi^{\mathrm{x}}_s) \in R^{\mathrm{x}}\}, \\ &\bar{R}^{\mathrm{x}}_s := \{x \,|\, (x, \xi^{\mathrm{x}}_s) \in \bar{R}^{\mathrm{x}}\},\ \bar{R}^{\mathrm{u}}_a := \{u \,|\, (u, \xi^{\mathrm{u}}_a) \in \bar{R}^{\mathrm{u}}\}. \end{aligned} \tag{9}$$

The following theorem transforms the original conditions of approximate bisimulation into a collection of set inclusions.

Theorem 1 *Let Σ be a discrete-time system and let $\mathcal{M} = (M^{\mathrm{x}}, M^{\mathrm{u}})$ a mesh. A binary relation R^{x} is an approximate bisimulation between Σ and the quantizer-embedding $\hat{\Sigma} = \mathrm{QE}(\Sigma, \mathcal{M})$ if the following conditions hold:*
1') for all $s \in \mathcal{S}_0$,

$$R^{\mathrm{x}}_s \supseteq C^{\mathrm{x}}_s. \tag{10}$$

2') for all $s \xrightarrow{a} s'$,

$$f^{\phi}_s(R^{\mathrm{x}}_s, C^{\mathrm{u}}_a) \subseteq R^{\mathrm{x}}_{s'}, \tag{11a}$$

$$\phi_s(R^{\mathrm{x}}_s) + C^{\mathrm{u}}_a \subseteq \bar{R}^{\mathrm{u}}_a \tag{11b}$$

where $f^{\phi}_s(x, u) = f(x, u + \phi(x, \xi^{\mathrm{x}}_s))$, $\phi_s(x) = \phi(x, \xi^{\mathrm{x}}_s)$.
3') for all $s \in \mathcal{S}$,

$$R^{\mathrm{x}}_s \subseteq \bar{R}^{\mathrm{x}}_s. \tag{12}$$

Proof: Using the interface for initial states, Condition 1 in Definition 1 is expressed as:

$$(x, Q^{\mathrm{x}}(x)) \in R^{\mathrm{x}}_0 \ \text{ for any } x \in X_0,$$

which is equivalent to Condition 1'.

Next, (11a)(11b) is rewritten as

$$\begin{aligned} &\forall (x, \hat{x}) \in R^{\mathrm{x}},\ u' \in \mathrm{dom}(M^{\mathrm{u}}), \\ &\begin{cases} (\, f(x, u' + \phi(x, \hat{x})), Q^{\mathrm{x}} \circ f(\hat{x}, Q^{\mathrm{u}}(u'))\,) \in R^{\mathrm{x}}, \\ (\, u' + \phi(x, \hat{x}), Q^{\mathrm{u}}(u')\,) \in \bar{R}^{\mathrm{u}}. \end{cases} \end{aligned} \tag{13}$$

Since $U \subset \mathrm{dom}(M^{\mathrm{u}})$, this immediately implies

$$\begin{aligned} &\forall (x, \hat{x}) \in R^{\mathrm{x}},\ \hat{u} \in Q^{\mathrm{u}}(U), \\ &\begin{cases} (\, f(x, \hat{u} + \phi(x, \hat{x})), Q^{\mathrm{x}} \circ f(\hat{x}, \hat{u})\,) \in R^{\mathrm{x}}, \\ (\, \hat{u} + \phi(x, \hat{x}), \hat{u}\,) \in \bar{R}^{\mathrm{u}}. \end{cases} \end{aligned} \tag{14}$$

Next, define $u = u' + \phi(x, \hat{x})$. Then, $u' = u + \phi(\hat{x}, x)$. Since we assume the input mesh is sufficiently large, the range of u given by $u = u' + \phi(x, \hat{x})$, $u' \in \mathrm{dom}(M^{\mathrm{u}})$, includes U. Thus we have

$$\begin{aligned} &\forall (x, \hat{x}) \in R^{\mathrm{x}},\ u \in U, \\ &\begin{cases} (\, f(x, u), Q^{\mathrm{x}} \circ f(\hat{x}, Q^{\mathrm{u}}(u + \phi(\hat{x}, x)))\,) \in R^{\mathrm{x}}, \\ (\, u, Q^{\mathrm{u}}(u + \phi(\hat{x}, x))\,) \in \bar{R}^{\mathrm{u}}. \end{cases} \end{aligned} \tag{15}$$

From (14) and (15), we obtain Condition 2. Finally, Condition 3' is obviously equivalent to Condition 3. This concludes the proof. ■

Theorem 1 still involves nonlinear set transformations, which are generally quite difficult to handle. In the next step, we recast these conditions in terms of errors between the states and inputs of the two systems. First, we restrict our attention to finding $\{R^{\mathrm{x}}_s\}$ that are expressed in the following specific form:

$$R^{\mathrm{x}}_s = \{x \,|\, \|x - \xi^{\mathrm{x}}_s\| \le \delta^{\mathrm{x}}_s\}. \tag{16}$$

Here, $\|\cdot\|$ denotes a norm, and δ^{x}_s is a positive constant. The variable δ^{x}_s describes how much the state x of Σ can deviate from the state $\hat{x}$ of $\hat{\Sigma}$. Given a norm, R^{x}_s is parameterized by δ^{x}_s. Moreover, let us denote by $\underline{r}(C, p)$ the maximum radius of a ball centered at p included in a set C. Similarly, let us denote by $\overline{r}(C, p)$ the minimum radius of a ball centered at p that includes C. Furthermore, we introduce a mild assumption on error specification. Let us define the following functions:

$$\begin{aligned} &A_i(x, u) = \left[\frac{\partial f_i}{\partial x} + \frac{\partial f_i}{\partial u}\frac{\partial \phi}{\partial x}\right](x, u), \\ &B_i(x, u) = \left[\frac{\partial f_i}{\partial u}\right](x, u),\ F_i(x) = \left[\frac{\partial \phi_i}{\partial x}\right](x). \end{aligned} \tag{17}$$

Moreover, let $A(x, u)$, $B(x, u)$ and $F(x)$ be functions that return matrices whose i-th rows are given by $A_i(x, u)$, $B_i(x, u)$ and $F_i(x)$, respectively.

Assumption 1 *There exists a constant $\kappa \geq 1$ such that the following holds: for any $(x, \hat{x}, u, \hat{u})$ such that $(x, \hat{x}) \in \bar{R}^{\mathrm{x}}$ and $(u, \hat{u}) \in \bar{R}^{\mathrm{u}}$,*

$$\|A'\| \leq \kappa \|A(\hat{x}, \hat{u})\|,\ \|B'\| \leq \kappa \|B(\hat{x}, \hat{u})\|,\ \|F'\| \leq \kappa \|F(\hat{x})\|.$$

Here, the i-th row of A', B' and F' are given by $A_i(x', u')$, $B_i(x', u')$ and $F_i(x', u')$, where $x' = \hat{x} + (x - \hat{x})\theta_i$ and $u' = \hat{u} + (u - \hat{u})\theta_i$ for some $\theta_i \in [0, 1]$. Moreover, norms for matrices are induced norms.

This assumption states that, for state-input pairs that are equivalent under the error specification $(\bar{R}^{\mathrm{x}}, \bar{R}^{\mathrm{u}})$, local linear systems obtained around their neighborhood should not differ significantly. For linear systems, one can simply set κ as 1.

The next theorem gives a sufficient condition for approximate bisimulation.

Theorem 2 *Let $\{\delta_s^{\mathrm{x}}\}$ $(s \in \mathcal{S})$ be a set of variables satisfying the following conditions:*
1") for all $s \in \mathcal{S}_0$,

$$\delta_s^{\mathrm{x}} \geq \overline{r}(C_s^{\mathrm{x}}, \xi_s^{\mathrm{x}}), \tag{18}$$

2") for all $s \xrightarrow{a} s'$,

$$\|A\|_{s,a}\, \delta_s^{\mathrm{x}} + \|B\|_{s,a}\, \delta_a^{\mathrm{u}} + \|f(\xi_s^{\mathrm{x}}, \xi_a^{\mathrm{u}}) - \xi_{s'}^{\mathrm{x}}\| \leq \delta_{s'}^{\mathrm{x}}, \tag{19a}$$
$$\|F\|_s\, \delta_s^{\mathrm{x}} + \delta_a^{\mathrm{u}} \leq \underline{r}(\bar{R}_a^{\mathrm{u}}, \xi_a^{\mathrm{u}}) \tag{19b}$$

where $\delta_a^{\mathrm{u}} = \overline{r}(C_a^{\mathrm{u}}, \xi_a^{\mathrm{u}})$, and
3") for all $s \in \mathcal{S}$,

$$\delta_s^{\mathrm{x}} \leq \underline{r}(\bar{R}_s^{\mathrm{x}}, \xi_s^{\mathrm{x}}). \tag{20}$$

Here,

$$\|A\|_{s,a} = \kappa \|A(\xi_s^{\mathrm{x}}, \xi_a^{\mathrm{u}})\|,\ \|B\|_{s,a} = \kappa \|B(\xi_s^{\mathrm{x}}, \xi_a^{\mathrm{u}})\|,$$
$$\|F\|_s = \kappa \|F(\xi_s^{\mathrm{x}})\|.$$

Then, a binary relation R^{x} given by (9) and (16) is an approximate bisimulation between Σ and $\hat{\Sigma}$.

Proof: The conditions (18) and (20) directly imply (10) and (12), respectively. Let $x \in R_s^{\mathrm{x}}$ and $u \in C_a^{\mathrm{u}}$. From the mean value theorem, we have

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

Here, the i-th row of A' and B' are given by $A_i(x', u')$ and $B_i(x', u')$, respectively, where $x' = \xi_s^{\mathrm{x}} + (x - \xi_s^{\mathrm{x}})\theta_i$, $u' = \xi_a^{\mathrm{u}} + (u - \xi_a^{\mathrm{u}})\theta_i$ with some $\theta_i \in [0, 1]$. From Assumption 1, $\|A'\| \leq \|A\|_{s,a}$ and $\|B'\| \leq \|B\|_{s,a}$. Therefore we have:

$$\begin{aligned}
&\|f_s^{\phi}(x, u) - \xi_{s'}^{\mathrm{x}}\| \\
&\quad = \|f(\xi_s^{\mathrm{x}}, \xi_a^{\mathrm{u}}) + A\,(x - \xi_s^{\mathrm{x}}) + B\,(u - \xi_a^{\mathrm{u}}) - \xi_{s'}^{\mathrm{x}}\| \\
&\quad \leq \|f(\xi_s^{\mathrm{x}}, \xi_a^{\mathrm{u}}) - \xi_{s'}^{\mathrm{x}}\| + \|A\|_{s,a}\delta_s^{\mathrm{x}} + \|B\|_{s,a}\delta_a^{\mathrm{u}} \leq \delta_{s'}^{\mathrm{x}},
\end{aligned}$$

thus $f_s^{\phi}(x, u) \in R_{s'}^{\mathrm{x}}$. Therefore, (19a) implies (11a). From a similar discussion, we can show that (19b) implies (11b). Therefore, from Theorem 1, R^{x} is an approximate bisimulation. ■

In Theorem 2, equation (19a) describes how the accumulated error of one symbolic state is propagated to other symbolic states. The term $\|A\|_{s,a}\, \delta_s^{\mathrm{x}}$ expresses the error of s being propagated to its successor s', the term $\|B\|_{s,a}\, \delta_a^{\mathrm{u}}$ expresses the input quantization error, and the term $\|f(\xi_s^{\mathrm{x}}, \xi_a^{\mathrm{u}}) - \xi_{s'}^{\mathrm{x}}\|$ expresses the state quantization error. Therefore, we can tell from Theorem 2 that approximate bisimulation is achieved if there exists a certain error distribution over the states of the discrete model that does not violate the error specification and satisfies the error propagation condition. In particular, an essential necessary condition for approximate bisimulation is that there does not exist a cycle on the discrete abstract model whose "loop gain" (the product of $\|A\|_{s,a}$s along the cycle) is greater than 1.

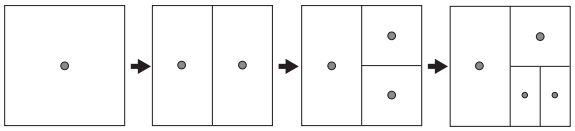


Fig. 2. An example of variable-resolution mesh

V. Computational methods for verification and design of Discrete Abstractions

In the previous section, we have derived a condition on a mesh that defines a quantizer-embedding satisfying a given error specification. Based on this result, in this section, we present practical methods for the verification and design of discrete abstract models.

A. Variable-resolution Quantizers

In this subsection, we characterize the notion of variable-resolution mesh. A variable-resolution mesh is a mesh with refinement operation. Refinement of a mesh is to partition a specified cell into two sub-cells. Moreover, it should be possible, by a proper sequence of refinements, to make the quantization error at any specified point arbitrarily small. Basically any type of meshes satisfying this requirement is applicable to the discrete abstraction design. One typical example of variable-resolution mesh is a mesh in which each cell is an orthogonal hyper-rectangle (Fig. 2). Initially, the mesh is composed of a single cell representing the entire domain whose node is place at the origin. A subdivision can be made in one of n directions. For an example, in the 2-dimensional case, a cell can be divided either horizontally or vertically. Each of newly created cells has its node in its middle.

B. Iterative Mesh Refinement Algorithm

Before proceeding to the design problem, we discuss how to verify whether a given mesh defines a discrete abstraction under a given error specification. A verification method is obtained almost directly based on Lemma **??**. Notice that the conditions given in Lemma **??** are expressed by a set of linear inequalities. In fact, each condition given in (18)-(20) is expressed in a general form of

$$\theta \leq \mathsf{A}x \leq \Theta. \tag{21}$$

TABLE I

PARAMETERS AND VARIABLES OF LINEAR INEQUALITIES

	x	A	θ	Θ
$\forall s \in \mathcal{S}_0$	$\delta^{\mathrm{x}}_{s,0}$	I	$\overline{r}(C^{\mathrm{x}}_s, \xi^{\mathrm{x}}_s)$	∞
$\forall a \in \mathcal{A}$	δ^{u}_a	I	$\overline{r}(C^{\mathrm{u}}_a, \xi^{\mathrm{u}}_a)$	∞
$\forall (s \xrightarrow{a} s')$	$\begin{bmatrix}\delta^{\mathrm{x}}_{s'}\\ \delta^{\mathrm{x}}_s\\ \delta^{\mathrm{u}}_a\end{bmatrix}$	$\begin{bmatrix}1\\ -\|A\|_{s,a}\\ -\|B\|_{s,a}\end{bmatrix}^{\mathrm{T}}$	$\begin{matrix}\|f(\xi^{\mathrm{x}}_s, \xi^{\mathrm{u}}_a)\\ -\xi^{\mathrm{x}}_{s'}\|\end{matrix}$	∞
	$\begin{bmatrix}\delta^{\mathrm{x}}_s\\ \delta^{\mathrm{u}}_a\end{bmatrix}$	$\begin{bmatrix}\|F\|_s\\ I\end{bmatrix}^{\mathrm{T}}$	$-\infty$	$\underline{r}(\bar{R}^{\mathrm{u}}_a, \xi^{\mathrm{u}}_a)$
$\forall s \in \mathcal{S}$	δ^{x}_s	I	$-\infty$	$\underline{r}(\bar{R}^{\mathrm{x}}_s, \xi^{\mathrm{x}}_s)$

Here, x denotes a variable or a vector of variables, A denotes a scalar constant or a constant row vector, θ denotes a lower bound and Θ an upper bound. Precise mapping between the actual symbols that appear in each condition and those used in (21) is summarized in Table I. Therefore, all the atomic conditions (18)-(20) can be aggregated into a single large linear inequality, which is also of the form (21) except that A is now a matrix. As a result, the problem of finding an approximate bisimulation is formulated as a feasibility verification problem of a set of linear inequalities. This problem can be solved numerically by means of a linear programming (LP) solver. Moreover, the inequality (21) has a clear structure; the matrix A is determined solely by system's dynamics, the lower bound θ is determined by the mesh resolution and the upper bound Θ comes from the error specification. Therefore, each component of θ can be reduced by a refinement of the corresponding cell. On the other hand, A and Θ cannot be directly manipulated.

Now, Let us proceed one step further and consider how to design a feasible mesh (a mesh satisfying (21)). From a practical perspective, we would like to have a discrete abstract model composed of as small number of symbolic states as possible; in other words, we would like a mesh not only to be feasible but also to be as coarse as possible. To achieve this goal, 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 efficiently. For this purpose, we need some continuous measure of how far the current mesh is from feasibility. Now, let us add an extra variable to (21) to obtain the following LP problem:

$$\mathrm{LP}(\mathcal{M}) \quad \begin{array}{ll} \text{minimize} & y \\ \text{sub. to} & x \geq \mathbf{0},\ y \geq 0, \\ & \theta \leq \mathsf{A}x + \theta y,\ \mathsf{A}x \leq \Theta. \end{array} \tag{22}$$

This LP has a new decision variable y. If y is seen as a parameter, LP($\mathcal{M}$) becomes equivalent to (21) when $y = 0$. Therefore, if the optimal cost of the LP($\mathcal{M}$) is 0, the mesh is feasible. Otherwise, the mesh does not have high enough resolution and thus needs further refinement. A question here is which portion of the mesh should be refined. To answer this question, we focus on the *dual variables* of LP($\mathcal{M}$). It is well known in the field of linear programming that the optimal solution of the dual problem can be used to measure the sensitivity of the minimum objective value against the change of constraint parameters. Let λ and Λ be the the dual variables corresponding to the inequality constraints $\theta \leq \mathsf{A}x + \theta y$ and $\mathsf{A}x \leq \Theta$, respectively. Then the dual problem of (22) is given as

$$\begin{array}{ll} \text{maximize} & \theta^{\mathrm{T}}\lambda - \Theta^{\mathrm{T}}\Lambda \\ \text{sub. to} & \lambda \geq \mathbf{0},\ \Lambda \geq \mathbf{0}, \\ & \mathsf{A}^{\mathrm{T}}(\lambda - \Lambda) \leq \mathbf{0}. \end{array} \tag{23}$$

When the primal problem is feasible, the dual problem is also feasible and the optimal objective values of these two problems coinside; we have

$$y^* = \theta^{\mathrm{T}}\lambda^* - \Theta^{\mathrm{T}}\Lambda^* \tag{24}$$

where $(\cdot)^*$ denotes the optimal value. We observe in this equation that the influence of θ_i, the i-th element of θ, on the optimal cost is measured by the product of θ_i and the corresponding dual variable λ^*_i. Coming back to Table I, θ_i is one of the following three terms: $\overline{r}(C^{\mathrm{x}}_s, \xi^{\mathrm{x}}_s)$, $\overline{r}(C^{\mathrm{u}}_a, \xi^{\mathrm{u}}_a)$ or $\|f(\xi^{\mathrm{x}}_s, \xi^{\mathrm{u}}_a) - \xi^{\mathrm{x}}_{s'}\|$. In the first two cases, θ_i is simply the radius of a cell and therefore it can be reduced monotonically by subdividing the corresponding cell. There is one difficulty in the third case, where $\theta_i = \|f(\xi^{\mathrm{x}}_s, \xi^{\mathrm{u}}_a) - \xi^{\mathrm{x}}_{s'}\|$; subdividing the cell $C^{\mathrm{x}}_{s'}$ does not always reduce θ_i. For this case, we make use of the fact that θ_i is upper-bounded by the radius of $C^{\mathrm{x}}_{s'}$; $\overline{r}(C^{\mathrm{x}}_{s'}, \xi^{\mathrm{x}}_{s'})$, which can always be reduced by subdividing $C^{\mathrm{x}}_{s'}$. Now we come to the following refinement rule; subdivide the *bottleneck cell*, which is a cell of either M^{x} or M^{u} such that the product of the change in its radius by a subdivision and the value of corresponding dual variable takes the largest value.

Based on the above discussion, we obtain the *Iterative Refinement Algorithm*, listed in Table II. The algorithm is provided with a system Σ, an error specification $(\bar{R}^{\mathrm{u}}, \bar{R}^{\mathrm{u}})$ and a constant ϵ specifying the minimum resolution. First, the algorithm initializes each of the mesh pair with a single cell covering the whole region. At every iteration, it constructs a quantizer-embedding using the current mesh. More precisely, the algorithm computes symbolic state transition relations, the coefficient matrices of a local linear system for each symbolic state-input pair (s, a) as well as their induced norms. The algorithm then constructs LP($\mathcal{M}$), where $\mathcal{M} = (M^{\mathrm{x}}, M^{\mathrm{u}})$. Any solver can be used for solving the LP, as long as it provides access to the values of dual variables at the optimal solution. As discussed earlier, if the optimal cost of LP($\mathcal{M}$) is 0, the mesh is verified and therefore it is output without further refinements. Otherwise, the algorithm refines the mesh. Here, (ξ, C) denotes the bottleneck cell. Superscripts ($^{\mathrm{x}}$ or $^{\mathrm{u}}$) and subscripts ($_s$ or $_a$) are omitted since it could be a cell of either M^{x} or M^{u}. If the size of the cell C, $\overline{r}(C, \xi)$, is smaller than ϵ, the algorithm terminates the refinement process and outputs $\emptyset$, indicating that the algorithm failed to construct a proper mesh.

TABLE II

ITERATIVE REFINEMENT ALGORITHM

Algorithm **refine_mesh**

Inputs

Σ	system
$(\bar{R}^{\mathrm{x}}, \bar{R}^{\mathrm{u}})$	error specification
ϵ	miminum resolution

Outputs

$(M^{\mathrm{x}}, M^{\mathrm{u}})$	mesh

$M^{\mathrm{x}} := \{(\mathbf{0}, \mathrm{dom}(M^{\mathrm{x}})\}$, $M^{\mathrm{u}} := \{(\mathbf{0}, \mathrm{dom}(M^{\mathrm{u}}))\}$
loop
  create quantizer embedding $\hat{\Sigma} = \mathrm{QE}(\Sigma, \mathcal{M})$
  solve LP($\mathcal{M}$).
  if $y^* = 0$,
   return $\mathcal{M}$.
  end
  $(\xi, C) :=$ the bottleneck cell.
  if $\bar{r}(C, \xi) < \epsilon$
   return $\emptyset$.
  end
  refine mesh at (ξ, C).
end

VI. NUMERICAL RESULTS

This section shows a simple example. Consider the following 2-dimensional linear system:

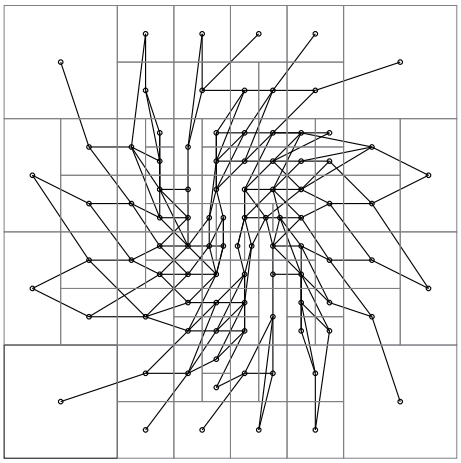
$$x_{t+1} = Ax_t + Bu_t, \quad A = \begin{bmatrix} 0.68 & -0.14 \\ 0.14 & 0.68 \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}.$$

The state set is given as $X = [-1, 1] \times [-1, 1]$ and the input set is given as $U = [-1, 1]$. For this system, we compute a discrete abstraction using the error specification $\bar{R}^{\mathrm{x}} = \{(x, \hat{x}) \,|\, \|x - \hat{x}\| \leq 0.25 + 0.5\|\hat{x}\|\}$. We assume common control inputs are applied to both systems and no error specification is imposed on them. The result is shown in Fig. 3 (a),(b). The discrete abstraction obtained is composed of 94 states and 16 inputs.

Fig. 4 shows 300 sample error histories of the state variables. Initial states and input trajectories are sampled uniformly at random. The horizontal axis represents the time interval $t \in [0 : 10]$ and the vertical axis plots the normalized error; $\|x_t - \hat{x}_t\|/(0.25 + 0.5\|\hat{x}_t\|)$. We can see that the normalized error is below 1.0 at every time instant, indicating that the error specification is satisfied.

VII. CONCLUSION

In this paper, we have presented a computational method for the discrete abstraction of nonlinear systems. First, it has been shown that the conditions of approximate bisimulation is transformed into a set of linear inequalities, which can be verified 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. In the future, the algorithm should be provided with a proof that it always outputs a discrete abstract model if the original system satisfies a certain existence condition.



(a) State mesh

(b) Input mesh

Fig. 3. Discrete abstraction of a linear system

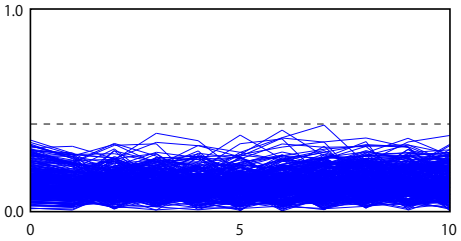


Fig. 4. Error histories of the discrete model and the concrete model

REFERENCES

[1] A. Girard and G.J. Pappas : Approximation metrics for discrete and continuous systems, IEEE Transactions on Automatic Control, 52(5), 782/798, 2007.

[2] A.A. Julius and G.J. Pappas: Approximate Equivalence and Approximate Synchronization of Metric Transition Systems, in the Proc. 45th IEEE Conf. Decision and Control 2006, San Diego, USA.

[3] A. Girard : Approximately Bisimilar Finite Abstractions of Stable Linear Systems, Hybrid Systems: Computation and Control, vol 4416 in LNCS, 231/244, Springer, 2007.

[4] 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.

[5] 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.

[6] A. Girard and G.J. Pappas : Hierarchical control system design using approximate simulation, Automatica, 45(2), 566/571, 2009.

[7] Y. Tazaki and J. Imura : Discrete-State Abstractions of Nonlinear Systems Using Multi-resolution Quantizer, 12th International Conference on Hybrid Systems: Computation and Control (HSCC'09), pp.351-365, San Francisco, CA, USA, April, 2009.