Discrete-state Abstractions of Nonlinear Systems Using Multi-resolution Quantizer

Yuichi Tazaki and Jun-ichi Imura

Tokyo Institute of Technology,
Ōokayama 2-12-1, Meguro, Tokyo, Japan
{tazaki,imura}@cyb.mei.titech.ac.jp
http://www.cyb.mei.titech.ac.jp

Abstract. This paper proposes a design method for discrete abstractions of nonlinear systems using multi-resolution quantizer, which is capable of handling state dependent approximation precision requirements. To this aim, we extend the notion of quantizer embedding, which has been proposed by the authors' previous works as a transformation from continuous-state systems to discrete-state systems, to a multi-resolution setting. Then, we propose a computational method that analyzes how a locally generated quantization error is propagated through the state space. Based on this method, we present an algorithm that generates a multi-resolution quantizer with a specified error precision by finite refinements. Discrete abstractions produced by the proposed method exhibit non-uniform distribution of discrete states and inputs.

1 Introduction

The problem of deriving a finite automaton that abstracts a given continuous-state system is called the discrete abstraction problem. A finite-state system is suitable for an abstract model since various difficult properties of continuous-state systems: nonlinearity, discontinuity, and non-convexity, ... can be handled in a uniform manner in a symbolic space. Until today, various techniques of discrete abstraction has been developed, and many of them are based on partitioning of state space. In [1][2][7], conditions for partitions that define discrete abstractions with deterministic transitions are discussed. On the other hand, in [3][4], instead of considering discrete abstractions of the open-loop behavior of continuous systems, a hierarchical controller composed of a symbolic transition system and a feedback controller that moves the continuous state to one region to another is proposed. There have also been much attention paid on building a symbolic system whose behavior includes the behavior of a continuous system. Such symbolic abstractions have been used for verification problems in [6], and for supervisory control in [8][9].

Recently, discrete abstraction methods based on approximate bisimulation [10] 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 abstraction of a wide range of systems can be obtained based on approximate bisimulation. In [11], a procedure for constructing approximately bisimilar finite abstractions of stable discrete-time linear systems has been derived. In [12] and [13], it has been shown that a general nonlinear system with the so-called incremental stability property can be abstracted by a uniform grid. The authors have also investigated the application of approximate bisimilar discrete abstractions to optimal control of linear systems with non-convex state constraints in [14], and to interconnected systems in [15]. 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. To date, however, it has the following limitations. First, the error condition in the conventional approximate bisimulation is uniform. From practical perspectives, it is desirable to support non-uniform error margin (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.

Motivated by the above backgrounds, this paper proposes a method for the design of discrete abstractions of nonlinear systems using multi-resolution quantizers. By using multi-resolution quantizers, one can design discrete abstract models with non-uniform distribution of states and inputs that approximate a given continuous-state system under state-dependent approximation precision requirements. Moreover, it also enables us to produce less conservative results compared to other conventional methods based on uniform discretization. To this end, in Section 2 we define the notion of finite-step abstraction. This notion admits an equivalence between two systems if any finite-step state trajectories of two systems generated with the same input signal satisfy a certain error criterion. In Section 3, we extend the notion of quantizer embedding, which has been presented in [14] and [15], to multi-resolution setting. This reduces the design of a discrete abstraction to the design of a pair of multi-resolution meshes, one is defined in the state space and another in the input space. In Section 4, we first discuss how to verify if a discrete model defined as a quantizer embedding with a given mesh satisfies the condition of the finite-step abstraction. This is basically done by computing how a locally generated quantization errors are propagated over the state space. Next, based on this verification method, we propose an algorithm that iteratively refines a multi-resolution mesh until the resultant quantizer-embedding is a finite-step abstraction of the original system. In Section 5, some illustrative examples are shown for demonstrating the effectiveness of the proposed method.

Notation: We write $[i_1 : i_2]$ to express the sequence of integers $i_1, i_1 + 1, \dots, i_2$. For an integer sequence $I = [i_1 : i_2]$, $\boldsymbol{u}_I = \{\boldsymbol{u}_{i_1}, \boldsymbol{u}_{i_1+1}, \dots, \boldsymbol{u}_{i_2}\}$. The symbol $\mathbb{R}$ denotes the field of real numbers and the symbol $\mathbb{Z}^+$ denotes the set of non-negative integers. For a vector $\boldsymbol{x} \in \mathbb{R}^n$, the symbol $\|\boldsymbol{x}\|_\infty$ denotes the ∞-norm of $\boldsymbol{x}$; $\|\boldsymbol{x}\|_\infty = \max_{i \in [1:n]} |\boldsymbol{x}_i|$.

2 Finite-step Abstraction of Discrete-time Systems

2.1 System description

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

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t). \tag{1}$$

We use the symbol U^N to express the set of N-step admissible control input sequences.

2.2 Finite-step abstraction

In the following, we introduce the notion of finite-step abstraction for the class of systems defined above.

Definition 1. *Finite-step abstraction*
Let $\Sigma\langle X, U, f\rangle$ and $\hat{\Sigma}\langle X, U, \hat{f}\rangle$ be discrete-time dynamical systems. Further, let $\bar{R} \subset X \times X$ be a binary relation between X and X. The system $\hat{\Sigma}$ is an N-step abstraction of Σ with respect to $\bar{R}$ if and only if for any initial state $\boldsymbol{x}_0 \in X$ of Σ, there exists an initial state $\hat{\boldsymbol{x}}_0 \in X$ of $\hat{\Sigma}$ such that the following holds:
for any $\boldsymbol{u}_{[0:N-1]} \in U^N$,

$$\begin{aligned} &(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t) \in \bar{R} \ \ (t \in [0:N]), \\ &\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t),\ \hat{\boldsymbol{x}}_{t+1} = \hat{f}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_t) \ \ (t \in [0:N-1]) \end{aligned} \tag{2}$$

holds.

The notion of finite-step abstraction can be seen as a variant of approximate bisimulation in the sense that:

i) it requires the similarity of trajectories for only a finite steps,
ii) it assumes common control inputs, whereas for approximate bisimulation two control inputs need not be the same, and
iii) the error condition is given in a more general form of binary relation $\bar{R}$, compared to the constant error bound of conventional approximate bisimulation.

The finite-step formulation could be a restriction. Nevertheless, it still has a wide range of potential applications. It can, of course, be used to solve problems that take place in a finite time interval, such as finite horizon optimal control. Moreover, it can be combined with model predictive control techniques to form a feedback-type controller. The relation $\bar{R}$ encodes an error condition imposed to two systems. The most simple example of $\bar{R}$ is a uniform error condition: $\bar{R} = \{(\boldsymbol{x}, \hat{\boldsymbol{x}}) \mid \|\boldsymbol{x} - \hat{\boldsymbol{x}}\| \le \epsilon\}$, where ϵ is a positive constant. On the other hand, $\bar{R}$ defined as $\bar{R} = \{(\boldsymbol{x}, \hat{\boldsymbol{x}}) \mid \|\boldsymbol{x} - \hat{\boldsymbol{x}}\| \le \eta\|\hat{\boldsymbol{x}}\| + \epsilon\}$, where both ϵ and η are positive constants, expresses a relative error condition.

3 Quantizer Embedding

In this section, we introduce the notion of *quantizer embedding*, which has been recently proposed by the authors in [15].

First of all, we introduce a *mesh* defined over a set $X \in \mathbb{R}^n$. A mesh $\mathcal{M}$ is a finite collection of pairs denoted by

$$\mathcal{M} = \{\{\boldsymbol{\xi}_0, C_0\}, \{\boldsymbol{\xi}_1, C_1\}, \ldots, \{\boldsymbol{\xi}_{\mathcal{S}}, C_{\mathcal{S}}\}\}. \tag{3}$$

Here, $\{C_0, C_1, \ldots, C_{\mathcal{S}}\}$ forms a partition of X; that is, $C_i \cap C_j = \emptyset \quad (i \neq j), \ \bigcup_i C_i = X$. Each $C_s \, (s \in [0 : \mathcal{S}])$ is called a *cell* of the mesh. Moreover, each $\boldsymbol{\xi}_s \in C_s$ is called a *discrete point* of the s-th cell. A mesh $\mathcal{M}$ defines a quantization function as shown below.

$$\begin{aligned} &Q[\mathcal{M}] \, : \, X \mapsto \{\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{\mathcal{S}}\}, \\ &Q[\mathcal{M}](\boldsymbol{x}) = \boldsymbol{\xi}_s \ \text{ if } \ \boldsymbol{x} \in C_s. \end{aligned} \tag{4}$$

A quantization function $Q[\mathcal{M}](\cdot)$ maps an arbitrary point $\boldsymbol{x}$ to a discrete point whose corresponding cell includes $\boldsymbol{x}$. We write $Q[\mathcal{M}](X) = \{\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{\mathcal{S}}\}$. Moreover, for any $\boldsymbol{x} \in X$, we write $Q[\mathcal{M}]^{-1}(\boldsymbol{x}) = C_s$ iff $Q[\mathcal{M}](\boldsymbol{x}) = \boldsymbol{\xi}_s$.

A discrete-time system can be 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 $\Sigma\langle X, U, f\rangle$ be a discrete-time system. Moreover let $Q^x(\cdot) := Q[\mathcal{M}^x](\cdot)$ be a quantizer defined in the state space and let $Q^u(\cdot) := Q[\mathcal{M}^u](\cdot)$ be a quantizer defined in the input space. The quantizer embedding (QE in short) of Σ, denoted by $\mathrm{QE}(\Sigma, Q^x, Q^u)$, is a system $\hat{\Sigma}\langle Q^x(X), U, \hat{f}\rangle$ whose state transition function is defined as

$$\hat{f}(\boldsymbol{x}, \boldsymbol{u}) := Q^x(f(\boldsymbol{x}, Q^u(\boldsymbol{u}))). \tag{5}$$

At every transition, the input of $\hat{\Sigma}$ is mapped to the discrete point of a cell of the input mesh $\mathcal{M}^u$ in which it is included. Moreover, the state of $\hat{\Sigma}$ is reset to the discrete point of a cell of the state mesh $\mathcal{M}^x$ in which the state right after a transition made by f is included. Therefore, as long as the meshes $\mathcal{M}^x$ and $\mathcal{M}^u$ are composed of a finite number of cells, a system with a state-transition of the form (5) can be viewed as a finite automaton. Once we assume that discrete models are expressed in terms of the quantizer embedding of the original system, the problem of discrete abstraction reduces to the design of a state mesh and an input mesh. From now on, to distinguish the cells and discrete points of the state mesh and the input mesh, we denote them by $\mathcal{M}^x = \{\{\boldsymbol{\xi}^x_s, C^x_s\}\}_{[0:\mathcal{S}]}$ and $\mathcal{M}^u = \{\{\boldsymbol{\xi}^u_a, C^u_a\}\}_{[0:\mathcal{A}]}$, respectively. The transition of $\hat{\Sigma}$ can be rewritten in a symbolic form as

$$s \xrightarrow{a} s' \ \Leftrightarrow \ f(\boldsymbol{\xi}^x_s, \boldsymbol{\xi}^u_a) \in C^x_{s'}. \tag{6}$$

Moreover, for later use, we define the *predecessor set* of a symbolic state s' as $\mathrm{pre}(s') = \{\{s, a\} \mid s \xrightarrow{a} s'\}$. Based on the above discussion, we formulate discrete abstraction as the following problem.

Problem 1. Discrete abstraction
For a discrete-time system $\Sigma\langle X, U, f\rangle$, $N \in \mathbb{Z}^+$ and a binary relation $\bar{R} \subset X \times X$, find a mesh $\mathcal{M}^x$ and $\mathcal{M}^u$ such that the quantizer-embedding $\mathrm{QE}(\Sigma, Q[\mathcal{M}^x], Q[\mathcal{M}^u])$ *is an N-step abstraction of Σ with respect to $\bar{R}$.*

4 Design of Multi-resolution Quantizer

4.1 Preparations

Throughout this section, we will make frequent use of interval operations. Interval computation technique has been used in reachability analysis problems (see [16]). It will be shown that this technique can also be utilized for the verification of discrete abstraction. Let $[x] = [\underline{x}, \overline{x}]$ and $[y] = [\underline{y}, \overline{y}]$ be closed intervals in $\mathbb{R}$. Elementary operations are defined as follows:

$$\begin{aligned}
[x] + [y] &= [\underline{x} + \underline{y},\ \overline{x} + \overline{y}],\\
[x] - [y] &= [\underline{x} - \overline{y},\ \overline{x} - \underline{y}],\\
[x][y] &= [\min\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\},\ \max\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}],\\
[x] \cup [y] &= [\min\{\underline{x}, \underline{y}\},\ \max\{\overline{x}, \overline{y}\}].
\end{aligned}$$

Moreover, $[x] \subseteq [y] \ \Leftrightarrow\ \underline{x} \geq \underline{y}, \overline{x} \leq \overline{y}$. Interval vectors and interval matrices are vectors and matrices whose elements are intervals. They obey the arithmetic rules of conventional matrices and vectors, except that element-wise operations are interval operations defined as above. We denote the set of all interval vectors of length n by $\mathbb{IR}^n$, and the set of all interval matrices with n rows and m columns by $\mathbb{IR}^{n\times m}$. The i-th element of an interval vector $[\boldsymbol{x}]$ is denoted by $[x_i] = [\underline{x}_i, \overline{x}_i]$. The element that lies in the i-th row and the j-th column of an interval matrix $[A]$ is denoted by $[A_{ij}] = [\underline{A}_{ij}, \overline{A}_{ij}]$. Let $[\boldsymbol{x}], [\boldsymbol{y}] \in \mathbb{IR}^n$ and let $\boldsymbol{a} \in \mathbb{R}^n$. We write

$$\begin{aligned}
\boldsymbol{a} \in [\boldsymbol{x}] &\ \Leftrightarrow\ a_i \in [x_i]\ \ \forall i \in [1:n],\\
[\boldsymbol{x}] \subseteq [\boldsymbol{y}] &\ \Leftrightarrow\ [x_i] \subseteq [y_i]\ \ \forall i \in [1:n].
\end{aligned}$$

Moreover, for compact sets $C \subset \mathbb{R}^n$ and $D \subset \mathbb{R}^n$, we define

$$(C, D) = \max_{\boldsymbol{a}\in C, \boldsymbol{b}\in D} \boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}.$$

For interval vectors, we have

$$([\boldsymbol{x}], [\boldsymbol{y}]) = \sum_{i\in[1:n]} \max_{a_i\in[x_i], b_i\in[y_i]} a_i b_i = \sum_{i\in[1:n]} \max\{\underline{x}_i\underline{y}_i, \underline{x}_i\overline{y}_i, \overline{x}_i\underline{y}_i, \overline{x}_i\overline{y}_i\}.$$

Some useful properties of the above operations are listed below for later use.

$$\begin{aligned}
&([\boldsymbol{x}] + [\boldsymbol{y}], \boldsymbol{v}) = ([\boldsymbol{x}], \boldsymbol{v}) + ([\boldsymbol{y}], \boldsymbol{v}) && ([\boldsymbol{x}], [\boldsymbol{y}] \in \mathbb{IR}^n, \boldsymbol{v} \in \mathbb{R}^n),\\
&([\boldsymbol{x}] \cup [\boldsymbol{y}], [\boldsymbol{z}]) = \max\{([\boldsymbol{x}], [\boldsymbol{z}]), ([\boldsymbol{y}], [\boldsymbol{z}])\} && ([\boldsymbol{x}], [\boldsymbol{y}], [\boldsymbol{z}] \in \mathbb{IR}^n),\\
&([A][\boldsymbol{x}], \boldsymbol{e}_i) = ([\boldsymbol{x}], [A]^{\mathrm{T}}\boldsymbol{e}_i) && ([\boldsymbol{x}] \in \mathbb{IR}^n, [A] \in \mathbb{IR}^{n\times n}).
\end{aligned}$$

Here, $\boldsymbol{e}_i$ is a vector whose i-th element is 1 and others are 0.

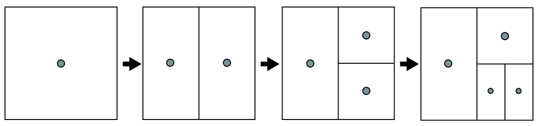


Fig. 1. Multi-resolution mesh

4.2 Multi-resolution quantizer

In this subsection, we introduce a class of quantizer defined by a multi-resolution mesh. First of all, we assume that the domain X of a quantizer is expressed as an interval vector of length n. A multi-resolution mesh takes the form of a binary tree, and each of its leaf nodes is assigned a cell of the mesh. Each cell is an interval vector of length n. Moreover, we assume that each discrete point is placed in the middle of the corresponding cell. Initially, the tree is composed of a single root node, whose cell represents the entire state set X. The tree can be grown by choosing an arbitrary leaf node, subdividing the corresponding cell into two sub-cells, and assigning each of them to one of two new child nodes that are created below the chosen node. A subdivision can be made in one of the n directions. For an example, in the 2-dimensional case, a cell can be divided either horizontally or vertically. Let $[C]_s$ be a cell expressed as $[C]_s = [[\underline{C}_{s,1}, \overline{C}_{s,1}], [\underline{C}_{s,2}, \overline{C}_{s,2}], \ldots, [\underline{C}_{s,n}, \overline{C}_{s,n}]]^{\mathrm{T}}$ and let $\boldsymbol{\xi}_s$ be the discrete point of $[C]_s$. Here, the i-th element of $\boldsymbol{\xi}_s$ is given by $\boldsymbol{\xi}_{s,i} = (\underline{C}_{s,i} + \overline{C}_{s,i})/2$. Subdividing $[C]_s$ in the i-th direction yields two new sub-cells: $[C]_{s_1} = [[\underline{C}_{s,1}, \overline{C}_{s,1}], \ldots, [\underline{C}_{s,i}, (\underline{C}_{s,i} + \overline{C}_{s,i})/2], \ldots, [\underline{C}_{s,n}, \overline{C}_{s,n}]]^{\mathrm{T}}$ and $[C]_{s_2} = [[\underline{C}_{s,1}, \overline{C}_{s,1}], \ldots, [(\underline{C}_{s,i} + \overline{C}_{s,i})/2, \overline{C}_{s,i}], \ldots, [\underline{C}_{s,n}, \overline{C}_{s,n}]]^{\mathrm{T}}$. The discrete points of the new cells are placed in the middle of them. Finally, since $[C]_s$ is no longer a leaf node after the subdivision, its cell and discrete point are removed from the mesh. Fig. 1 illustrates how a multi-resolution mesh is refined.

4.3 Verification of N-step abstraction

In this subsection, we discuss how to verify, for a given discrete-time system $\Sigma\langle X, U, f\rangle$ and a given pair of multi-resolution meshes $\mathcal{M}^x$ and $\mathcal{M}^u$, whether $\hat{\Sigma}$ defined as $\hat{\Sigma} = \mathrm{QE}(\Sigma, Q^x, Q^u)$ ($Q^x = Q[\mathcal{M}^x]$, $Q^u = Q[\mathcal{M}^u]$) is an N-step abstraction of Σ. From later on, we will write $\mathcal{M}^x = \{\{\boldsymbol{\xi}^x_s, [C]^x_s\}\}_{[0:\mathcal{S}]}$, $\mathcal{M}^u = \{\{\boldsymbol{\xi}^u_a, [C]^u_a\}\}_{[0:\mathcal{A}]}$, since the cells of the meshes are assumed to be interval vectors. Let us define the following sequence of binary relations:

$$R_0 = \bigcup_{\boldsymbol{x} \in X} (\boldsymbol{x}, Q^x(\boldsymbol{x})), \quad R_t = \bigcup_{(\boldsymbol{x}, \hat{\boldsymbol{x}}) \in R_{t-1},\, \boldsymbol{u} \in U} (f(\boldsymbol{x}, \boldsymbol{u}), \hat{f}(\hat{\boldsymbol{x}}, \boldsymbol{u})). \tag{7}$$

We will transform this recursive expression of R_t in such a way that the discrete-state characteristics of $\hat{\Sigma}$ is made more explicit. Let us define $R_{t,s} := \{\boldsymbol{x} \mid (\boldsymbol{x}, \boldsymbol{\xi}^x_s)$

$\in R_t\}$. Then (7) is rewritten using $R_{t,s}$ as follows:

$$R_{0,s} = [C]^x_s, \quad R_{t,s'} = \bigcup_{\{s,a\}\in \mathrm{pre}(s')} f(R_{t-1,s}, [C]^u_a). \tag{8}$$

Here, $f(R_{t-1,s}, [C]^u_a) = \{f(\boldsymbol{x}, \boldsymbol{u}) \,|\, \boldsymbol{x} \in R_{t-1,s}, \boldsymbol{u} \in [C]^u_a\}$. The following lemma provides the simplest way for checking the N-step abstraction.

Lemma 1. *$\hat{\Sigma}$ is an N-step abstraction of Σ if $R_t \subseteq \bar{R}$ holds for all $t \in [0 : N]$, or equivalently, if $R_{t,s} \subseteq \bar{R}_s$ holds for all $t \in [0 : N]$ and $s \in [0 : \mathcal{S}]$, where $\bar{R}_s = \{\boldsymbol{x} \,|\, (\boldsymbol{x}, \boldsymbol{\xi}^x_s) \in \bar{R}\}$.*

It is in general difficult to compute the nonlinear set operation $f(R_{t-1,s}, [C]^u_a)$. Moreover, due to the set union operations, $R_{t,s}$ may become highly non-convex as t increases. In the following, we derive a practical method that computes a conservative approximation of (8), taking advantage of interval computation techniques. The next lemma provides us with a conservative linear approximation of the nonlinear set operation $f(R_{t-1,s}, [C]^u_a)$.

Lemma 2. *Let C^x and C^u be convex subsets of X and U, respectively, and let $\boldsymbol{\xi}^x \in C^x$, $\boldsymbol{\xi}^u \in C^u$. Moreover, let $[A](C^x, C^u)$ and $[B](C^x, C^u)$ be functions that return interval matrices defined as*

$$[A_{ij}](C^x, C^u) = \left[\min_{\boldsymbol{x}\in C^x, \boldsymbol{u}\in C^u} \frac{\partial f_i}{\partial x_j}(\boldsymbol{x}, \boldsymbol{u}), \max_{\boldsymbol{x}\in C^x, \boldsymbol{u}\in C^u} \frac{\partial f_i}{\partial x_j}(\boldsymbol{x}, \boldsymbol{u})\right], \tag{9}$$

$$[B_{ij}](C^x, C^u) = \left[\min_{\boldsymbol{x}\in C^x, \boldsymbol{u}\in C^u} \frac{\partial f_i}{\partial u_j}(\boldsymbol{x}, \boldsymbol{u}), \max_{\boldsymbol{x}\in C^x, \boldsymbol{u}\in C^u} \frac{\partial f_i}{\partial u_j}(\boldsymbol{x}, \boldsymbol{u})\right]. \tag{10}$$

Then, the following inclusion holds:

$$f(C^x, C^u) \subseteq [A](C^x, C^u)(C^x - \boldsymbol{\xi}^x) + [B](C^x, C^u)(C^u - \boldsymbol{\xi}^u) + f(\boldsymbol{\xi}^x, \boldsymbol{\xi}^u). \tag{11}$$

Proof. From the mean value theorem, for any $\boldsymbol{x} \in C^x$, $\boldsymbol{u} \in C^u$ and $i \in [1 : n]$, there exists a $\theta \in [0, 1]$ such that

$$f_i(\boldsymbol{x}, \boldsymbol{u}) = f_i(\boldsymbol{\xi}^x, \boldsymbol{\xi}^u) + \frac{\partial}{\partial \boldsymbol{x}} f_i(\boldsymbol{x}', \boldsymbol{u}')(\boldsymbol{x} - \boldsymbol{\xi}^x) + \frac{\partial}{\partial \boldsymbol{u}} f_i(\boldsymbol{x}', \boldsymbol{u}')(\boldsymbol{u} - \boldsymbol{\xi}^u)$$

where $f_i(\boldsymbol{x}, \boldsymbol{u})$ denotes the i-th element of $f(\boldsymbol{x}, \boldsymbol{u})$, $\boldsymbol{x}' = \boldsymbol{\xi}^x + \theta(\boldsymbol{x} - \boldsymbol{\xi}^x)$ and $\boldsymbol{u}' = \boldsymbol{\xi}^u + \theta(\boldsymbol{u} - \boldsymbol{\xi}^u)$. From the convexity assumption, $\boldsymbol{x}' \in C^x$, $\boldsymbol{u}' \in C^u$. This means $(\partial/\partial\boldsymbol{x})f_i(\boldsymbol{x}', \boldsymbol{u}')$ is included in the i-th row interval vector of $[A](C^x, C^u)$ and $(\partial/\partial\boldsymbol{u})f_i(\boldsymbol{x}', \boldsymbol{u}')$ is included in the i-th row interval vector of $[B](C^x, C^u)$. This completes the proof. □

Now, let $[\mathcal{E}]_{t,s}$ be a sequence of interval vectors defined as

$$[\mathcal{E}]_{0,s} = [C]^x_s - \boldsymbol{\xi}^x_s, \tag{12}$$

$$[\mathcal{E}]_{t,s'} = \bigcup_{\{s,a\}\in \mathrm{pre}(s')} [[A]_{t-1,s,a}[\mathcal{E}]_{t-1,s} + [B]_{t-1,s,a}([C]^u_a - \boldsymbol{\xi}^u_a) + (f(\boldsymbol{\xi}^x_s, \boldsymbol{\xi}^u_a) - \boldsymbol{\xi}^x_{s'})] \tag{13}$$

where $[A]_{t,s,a} = [A]([\mathcal{E}]_{t,s} + \boldsymbol{\xi}^x_s, [C]^u_a)$ and $[B]_{t,s,a} = [B]([\mathcal{E}]_{t,s} + \boldsymbol{\xi}^x_s, [C]^u_a)$. From Lemma 2, we have $[\mathcal{E}]_{t,s} + \boldsymbol{\xi}^x_s \supseteq R_{t,s}$; thus, $[\mathcal{E}]_{t,s} + \boldsymbol{\xi}^x_s$ is an over-approximation of $R_{t,s}$. This means that $[\mathcal{E}]_{t,s}$ can be seen as a conservative estimate of the *accumulated error* between $\boldsymbol{x}_t$ and $\hat{\boldsymbol{x}}_t$ when $\hat{\boldsymbol{x}}_t = \boldsymbol{\xi}^x_s$ ($\hat{\boldsymbol{x}}_t$ denotes the state of $\hat{\Sigma}$ at time t). Furthermore, equation (13) describes how the accumulated error of one symbolic state is propagated to other symbolic states. The term $[A]_{t-1,s,a}[\mathcal{E}]_{t-1,s}$ expresses the error of s being propagated to its successor s', the term $[B]_{t-1,s,a}([C]^u_a - \boldsymbol{\xi}^u_a)$ expresses the input quantization error, and the term $(f(\boldsymbol{\xi}^x_s, \boldsymbol{\xi}^u_a) - \boldsymbol{\xi}^x_{s'})$ expresses the state quantization error. For later use, we write equation (12), (13) in the form of an algorithm.

```
**propagate_error**
    for each $s \in \mathcal{S}$ do $[\mathcal{E}]_{0,s} := [C]^x_s - \boldsymbol{\xi}^x_s$
    for $t = 1$ to $N$
        for each $s' \in \mathcal{S}$
            $[\mathcal{E}]_{t,s'} := \bigcup_{\{s,a\} \in \mathrm{pre}(s')} [[A]_{t-1,s,a}[\mathcal{E}]_{t-1,s} + [B]_{t-1,s,a}([C]^u_a - \boldsymbol{\xi}^u_a)$
                $+(f(\boldsymbol{\xi}^x_s, \boldsymbol{\xi}^u_a) - \boldsymbol{\xi}^x_{s'})]$
        end
    end
```

The next lemma gives a sufficient condition for the N-step abstraction.

Lemma 3. *$\hat{\Sigma}$ is an N-step abstraction of Σ if*

$$[\mathcal{E}]_{t,s} \subseteq \bar{\mathcal{E}}_s \tag{14}$$

holds for all $t \in [0 : N]$, $s \in [0 : \mathcal{S}]$, where $\bar{\mathcal{E}}_s = \bar{R}_s - \boldsymbol{\xi}^x_s$.

From later on, we focus on cases in which $\bar{R}_s$ is expressed as an interval vector. In such cases, the evaluation of $[\mathcal{E}]_{t,s} \subseteq \bar{\mathcal{E}}_s$ is done by comparing real values at most $2n$ times. The following algorithm checks the N-step abstraction.

```
**max_violation**
    $\{t, s, \mu, i\} := \mathrm{argmax}_{t \in [0:N], s \in [0:\mathcal{S}], \mu \in \{-1,1\}, i \in [1:n]}(([\mathcal{E}]_{t,s}, \mu \boldsymbol{e}_i) - (\bar{\mathcal{E}}_s, \mu \boldsymbol{e}_i))$
    $p := ([\mathcal{E}]_{t,s}, \mu \boldsymbol{e}_i) - (\bar{\mathcal{E}}_s, \mu \boldsymbol{e}_i)$
    return $\{t, s, \mu, i, p\}$
```

The **max_violation** algorithm returns a tuple $\{t, s, \mu, i, p\}$, in which t, s and $\mu \boldsymbol{e}_i$ denote the time, the cell index and the direction of the maximum error violation, respectively, and p denotes the corresponding amount of violation. If $p \leq 0$, it means that condition (14) is satisfied and therefore no further refinement of the meshes is needed. In the next subsection, we discuss how to refine the meshes if p returned by **max_violation** has a positive value.

4.4 Iterative refinement of meshes

Let us consider how to design a pair of meshes that satisfies (14). We want the meshes not only to satisfy (14), but also to be as coarse as possible. However, it is extremely difficult to guarantee that the obtained meshes are optimal in terms of the number of cells. Instead, we take an iterative and greedy approach; starting from a state mesh and an input mesh both composed of a single cell, we subdivide a cell in a certain direction one by one until the error condition (14) is satisfied. At each iteration, we first detect a cell and a direction that violate the error margin most significantly. This is done by calling **propagate_error** followed by **max_violation**, defined in the last subsection. Let us denote the return values of **max_violation** by $\{t, s', \mu, i, p\}$. Again, if $p \leq 0$ then no further refinement is needed and we can get out of the loop. Otherwise, we identify a cell-direction pair whose contribution to the error $([\mathcal{E}]_{t,s'}, \mu \boldsymbol{e}_i)$ is the largest. From (13), $([\mathcal{E}]_{t,s'}, \mu \boldsymbol{e}_i)$ is given by the following:

$$([\mathcal{E}]_{t,s'}, \mu \boldsymbol{e}_i) = \begin{cases} ([C]^x_{s'} - \boldsymbol{\xi}^x_{s'}, \mu \boldsymbol{e}_i) & \text{if } t = 0, \\ \max\limits_{\{s,a\} \in \mathrm{pre}(s')} \big(([\mathcal{E}]_{t-1,s}, \mu [A]^{\mathrm{T}}_{t-1,s,a} \boldsymbol{e}_i) + ([C]^u_a - \boldsymbol{\xi}^u_a, \mu [B]^{\mathrm{T}}_{t-1,s,a} \boldsymbol{e}_i) & \\ \qquad + (f(\boldsymbol{\xi}^x_s, \boldsymbol{\xi}^u_a) - \boldsymbol{\xi}^x_{s'}, \mu \boldsymbol{e}_i) \big) & \text{otherwise.} \end{cases} \tag{15}$$

For later use, let us denote by $\mathrm{pre}(t, s', \mu, i)$ the pair $\{s, a\}$ that gives the maximum in the right hand side of (15). We can observe from (15) that for $t = 0$, the only way to reduce $([\mathcal{E}]_{t,s'}, \mu \boldsymbol{e}_i)$ is to reduce $([C]^x_{s'} - \boldsymbol{\xi}^x_{s'}, \boldsymbol{e}_i)$; i.e., to subdivide the cell $[C]^x_{s'}$ in the i-th direction. On the other hand, for $t > 0$, $([\mathcal{E}]_{t,s'}, \mu \boldsymbol{e}_i)$ is given by the maximum of the sum of three terms, where the maximum is taken among all the predecessors of s'. Note that those three terms are interpreted as: the accumulated error propagated from s, the input quantization error, and the state quantization error. Therefore, we have three options to reduce $([\mathcal{E}]_{t,s'}, \mu \boldsymbol{e}_i)$: to reduce the accumulated error $([\mathcal{E}]_{t-1,s}, \mu [A]^{\mathrm{T}}_{t-1,s,a} \boldsymbol{e}_i)$, to reduce the input quantization error $([C]^u_a - \boldsymbol{\xi}^u_a, \mu [B]^{\mathrm{T}}_{t-1,s,a} \boldsymbol{e}_i)$, and to reduce the state quantization error $(f(\boldsymbol{\xi}^x_s, \boldsymbol{\xi}^u_a) - \boldsymbol{\xi}^x_{s'}, \mu \boldsymbol{e}_i)$.

To reduce $([C]^u_a - \boldsymbol{\xi}^u_a, \mu [B]^{\mathrm{T}}_{t-1,s,a} \boldsymbol{e}_i)$, we should subdivide $[C]^u_a$. However, a question arises; in which direction should $[C]^u_a$ be subdivided? Now, notice that $([C]^u_a - \boldsymbol{\xi}^u_a, \mu [B]^{\mathrm{T}}_{t-1,s,a} \boldsymbol{e}_i)$ can be further decomposed into the following form:

$$\begin{aligned} ([C]^u_a - \boldsymbol{\xi}^u_a, \mu [B]^{\mathrm{T}}_{t-1,s,a} \boldsymbol{e}_i) &= ([C], \mu [B^{\mathrm{row}}_i]^{\mathrm{T}}) = \sum_{j \in [1:m]} ([C_j], \mu [B_{ij}]) \\ &= \sum_{j \in [1:m]} \max\{([C_j], \mu \underline{B}_{ij}), ([C_j], \mu \overline{B}_{ij})\} \\ &= \sum_{j \in [1:m]} \max\{([C], \mu \underline{B}_{ij} \boldsymbol{e}_j), ([C], \mu \overline{B}_{ij} \boldsymbol{e}_j)\}. \end{aligned} \tag{16}$$

For ease of notation, we temporarily write $[C] = [C]^u_a - \boldsymbol{\xi}^u_a$. Moreover, $[B^{\mathrm{row}}_i]$ and $[B_{ij}] = [\underline{B}_{ij}, \overline{B}_{ij}]$ denote the i-th row vector and the (i, j)-element of $[B]_{t-1,s,a}$, respectively. From this equation, we observe that by subdividing $[C]^u_a$ in the j-th direction whose corresponding term in the summation is the largest, the overall input quantization error will be reduced the most. Thus, we choose this j as the direction of subdivision.

For reducing $(f(\boldsymbol{\xi}_s^x, \boldsymbol{\xi}_a^u) - \boldsymbol{\xi}_{s'}^x, \mu \boldsymbol{e}_i)$, it is natural to subdivide $[C]_{s'}^x$. However, we need an additional care; subdividing $[C]_{s'}^x$ in the i-th direction does not always reduce $(f(\boldsymbol{\xi}_s^x, \boldsymbol{\xi}_a^u) - \boldsymbol{\xi}_{s'}^x, \mu \boldsymbol{e}_i)$. If $(f(\boldsymbol{\xi}_s^x, \boldsymbol{\xi}_a^u) - \boldsymbol{\xi}_{s'}^x)^{\mathrm{T}} \boldsymbol{e}_i > 0$ then the change of $(f(\boldsymbol{\xi}_s^x, \boldsymbol{\xi}_a^u) - \boldsymbol{\xi}_{s'}^x)$ resulting from the subdivision in the i-th direction is given by $-(([C]_{s'}^x - \boldsymbol{\xi}_{s'}^x, \boldsymbol{e}_i)/2)\boldsymbol{e}_i$. On the other hand, if $(f(\boldsymbol{\xi}_s^x, \boldsymbol{\xi}_a^u) - \boldsymbol{\xi}_{s'}^x)^{\mathrm{T}} \boldsymbol{e}_i < 0$, the change of $(f(\boldsymbol{\xi}_s^x, \boldsymbol{\xi}_a^u) - \boldsymbol{\xi}_{s'}^x)$ will be $(([C]_{s'}^x - \boldsymbol{\xi}_{s'}^x, \boldsymbol{e}_i)/2)\boldsymbol{e}_i$. This means subdivision is effective if and only if $(f(\boldsymbol{\xi}_s^x, \boldsymbol{\xi}_a^u) - \boldsymbol{\xi}_{s'}^x)^{\mathrm{T}} \boldsymbol{e}_i$ and μ have the same sign.

Finally, let us consider reducing $([\mathcal{E}]_{t-1,s}, \mu [A]_{t-1,s,a}^{\mathrm{T}} \boldsymbol{e}_i)$. Like (16), we have

$$([\mathcal{E}]_{t-1,s}, \mu [A]_{t-1,s,a}^{\mathrm{T}} \boldsymbol{e}_i) = \sum_{j \in [1:n]} \max\{([\mathcal{E}]_{t-1,s}, \mu \underline{A}_{ij} \boldsymbol{e}_j), ([\mathcal{E}]_{t-1,s}, \mu \overline{A}_{ij} \boldsymbol{e}_j)\} \tag{17}$$

where $[\underline{A}_{ij}, \overline{A}_{ij}]$ denotes the (i,j)-element of $[A]_{t-1,s,a}$. But in this case, it is not straightforward to determine which cell (and in which direction) should be subdivided in order to reduce $([\mathcal{E}]_{t-1,s}, \mu \boldsymbol{e}_j)$ for given $\mu \in \mathbb{R}$ and $j \in [1:n]$. This is because $[\mathcal{E}]_{t-1,s}$ is itself an accumulated sum of quantization errors caused by all the predecessors of s. Hence, we shall repeat the same discussion as above in order to identify a cell and its direction to be subdivided. This leads us to the following recursive algorithm, which determines a cell-direction pair that has the most significant influence on $([\mathcal{E}]_{t,s'}, \mu \boldsymbol{e}_i)$.

```
identify_bottleneck($t, s', \mu, i$)
    if $t = 0$
        return $\{s', i, ([C]_{s'}^x - \boldsymbol{\xi}_{s'}^x, \mu \boldsymbol{e}_i)/2\}$
    else
        $\mathcal{C}_1 := \emptyset$, $\mathcal{C}_2 := \emptyset$, $\mathcal{C}_3 := \emptyset$
        $\{s, a\} := \mathrm{pre}(t, s', \mu, i)$
        $[A] := [A]([\mathcal{E}]_{t-1,s} + \boldsymbol{\xi}_s^x, [C]_a^u)$
        $[B] := [B]([\mathcal{E}]_{t-1,s} + \boldsymbol{\xi}_s^x, [C]_a^u)$
        if $\mathrm{sgn}((f(\boldsymbol{\xi}_s^x, \boldsymbol{\xi}_a^u) - \boldsymbol{\xi}_{s'}^x, \boldsymbol{e}_i)) = \mathrm{sgn}(\mu)$
            $\mathcal{C}_1 := \mathcal{C}_1 \cup \{s', i, ([C]_{s'}^x - \boldsymbol{\xi}_{s'}^x, \mu \boldsymbol{e}_i)/2\}$
        end
        for $j \in [1:m]$
            $\mathcal{C}_2 := \mathcal{C}_2 \cup \{a, j, ([C]_a^u - \boldsymbol{\xi}_a^u, \mu \underline{B}_{ij} \boldsymbol{e}_j)/2\}$
                     $\cup \{a, j, ([C]_a^u - \boldsymbol{\xi}_a^u, \mu \overline{B}_{ij} \boldsymbol{e}_j)/2\}$
        end
        for $j \in [1:n]$
            $\mathcal{C}_3 := \mathcal{C}_3 \cup$ identify_bottleneck($t-1, s, \mu \underline{A}_{ij}, j$)
                     $\cup$ identify_bottleneck($t-1, s, \mu \overline{A}_{ij}, j$)
        end
        return $\mathrm{argmax}_{\{c,d,p\} \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3}\, p$
    end
```

Let $\{c^*, d^*, p^*\}$ be the return values of **identify_bottleneck**(t, s', μ, i). Here, the variable c^* could contain either a state symbol or an input symbol. In the former case, $[C]^x_{c^*}$ is a cell of $\mathcal{M}^x$, and in the latter case, $[C]^u_{c^*}$ is a cell of $\mathcal{M}^u$. We may omit the superscript and write $[C]_{c^*}$ when the distinction is not necessary. Connecting the above components all together, we obtain the following algorithm.

```
refine_mesh
    $\mathcal{M}^x := \{\{\mathbf{0}, X\}\}$, $\mathcal{M}^u := \{\{\mathbf{0}, U\}\}$
    loop
        propagate_error
        $\{t, s', \mu, i, p\}$ := max_violation
        if $p \le 0$ then terminate
        $\{c^*, d^*, p^*\}$ := identify_bottleneck$(t, s', \mu, i)$
        subdivide$(c^*, d^*)$
    end
```

Here, **subdivide** is a procedure that divides the cell $[C]_{c^*}$ in the direction d^*.

In the following, we will show that the algorithm **refine_mesh** actually terminates in finite iterations and produces a pair of meshes that defines an N-step abstraction of the given original system. First, we introduce the following assumption:

Assumption 1 *There exists a positive constant ϵ that satisfies*

$$\{(\boldsymbol{x}, \hat{\boldsymbol{x}}) \mid \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_\infty \le \epsilon\} \subseteq \bar{R}. \tag{18}$$

This assumption is fairly natural and is satisfied in most cases including uniform and relative error conditions shown in Section 2. Moreover, let us define

$$\lambda_A = \max_{\boldsymbol{x}\in X, \boldsymbol{u}\in U, i\in[1:n], j\in[1:n]} \left| \frac{\partial f_i}{\partial x_j}(\boldsymbol{x}, \boldsymbol{u}) \right|,$$
$$\lambda_B = \max_{\boldsymbol{x}\in X, \boldsymbol{u}\in U, i\in[1:n], j\in[1:m]} \left| \frac{\partial f_i}{\partial u_j}(\boldsymbol{x}, \boldsymbol{u}) \right|.$$

The next lemma claims that there exists a lower-bound on the size of cells that are subdivided at each iteration of **refine_mesh**.

Lemma 4. *In **refine_mesh**, a pair (c^*, d^*) passed to **subdivide** at each iteration satisfies*

$$([C]_{c^*} - \boldsymbol{\xi}_{c^*}, \boldsymbol{e}_{d^*}) \ge \frac{\epsilon}{\alpha_N \beta_N} \tag{19}$$

where $\alpha_0 = \beta_0 = 1$ and $\alpha_t = \max\{\max_{k\in[0:t-1]} \lambda_A^k \lambda_B, \max_{k\in[0:t]} \lambda_A^k\}$, $\beta_t = (\sum_{k=0}^{t} n^k + m \sum_{k=0}^{t-1} n^k)$ for $t \ge 1$.

Proof. At first, we will prove by construction that the output of **identify_ bottleneck**(t, s', μ, i), denoted by $\{c^*, d^*, p^*\}$, satisfies

$$([\mathcal{E}]_{t,s'}, \mu \boldsymbol{e}_i) \leq 2p^* \beta_t, \tag{20}$$

$$([C]_{c^*} - \boldsymbol{\xi}_{c^*}, \boldsymbol{e}_{d^*}) \geq 2p^*/(|\mu|\alpha_t). \tag{21}$$

When $t = 0$, **identify_bottleneck** immediately returns $\{s', i, ([C]^x_{s'} - \boldsymbol{\xi}^x_{s'}, \mu \boldsymbol{e}_i)/2\}$. Here, $([\mathcal{E}]_{0,s'}, \mu \boldsymbol{e}_i) = ([C]^x_{s'} - \boldsymbol{\xi}^x_{s'}, \mu \boldsymbol{e}_i) = 2p^*$. Moreover, $([C]^x_{s'} - \boldsymbol{\xi}^x_{s'}, \boldsymbol{e}_i) = 2p^*/|\mu|$. Therefore we confirm that (20) and (21) hold for $t = 0$.

Now, suppose (20) and (21) hold for $t = \tau - 1$ and consider the case of $t = \tau$. Let us denote $\{c_\rho, d_\rho, p_\rho\} = \mathrm{argmax}_{\{c,d,p\} \in \mathcal{C}_\rho} p$ for each $\rho \in \{1, 2, 3\}$ (recall that $\mathcal{C}_\rho$ are temporary variables that appear in **identify_bottleneck**). Then, p^* is given by $p^* = \max\{p_1, p_2, p_3\}$. Here we have

$$(f(\boldsymbol{\xi}^x_s, \boldsymbol{\xi}^u_a) - \boldsymbol{\xi}^x_{s'}, \mu \boldsymbol{e}_i) \leq 2p_1, \;\; ([C]^u_a - \boldsymbol{\xi}^u_a, \mu [B]^{\mathrm{T}}_{t-1,s,a} \boldsymbol{e}_i) \leq 2mp_2.$$

Moreover, from (20) with $t = \tau - 1$ we have

$$([\mathcal{E}]_{\tau-1,s}, \mu [A]^{\mathrm{T}}_{\tau-1,s,a} \boldsymbol{e}_i) \leq 2np_3 \beta_{\tau-1}.$$

Therefore

$$([\mathcal{E}]_{\tau,s'}, \mu \boldsymbol{e}_i) \leq 2p_1 + 2mp_2 + 2np_3 \beta_{\tau-1} \leq 2p^* \beta_\tau.$$

On the other hand, we have

$$([C]^x_{s'} - \boldsymbol{\xi}^x_{s'}, \boldsymbol{e}_{d_1}) \geq 2p_1/|\mu|, \;\; ([C]^u_a - \boldsymbol{\xi}^u_a, \boldsymbol{e}_{d_2}) \geq 2p_2/(|\mu|\lambda_B),$$
$$([C]_{c_3} - \boldsymbol{\xi}_{c_3}, \boldsymbol{e}_{d_3}) \geq 2p_3/(|\mu|\lambda_A \alpha_{\tau-1}).$$

This yields

$$([C]_{c^*} - \boldsymbol{\xi}_{c^*}, \boldsymbol{e}_{d^*}) \geq 2p^*/\max\{|\mu|, \, |\mu|\lambda_B, \, |\mu|\lambda_A \alpha_{\tau-1}\} \geq 2p^*/(|\mu|\alpha_\tau),$$

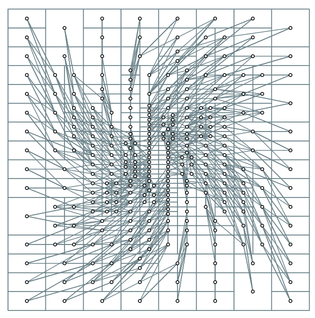
and we conclude that (20) and (21) hold for any $t \geq 0$.

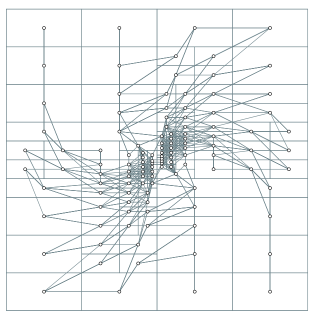
From Assumption 1, we have $(\bar{\mathcal{E}}_{s'}, \mu \boldsymbol{e}_i) \geq \epsilon$ for $\mu \in \{-1, 1\}$. This means $([\mathcal{E}]_{t,s'}, \mu \boldsymbol{e}_i) \geq \epsilon$ holds for a tuple $\{t, s', \mu, i\}$ passed to **identify_bottleneck**. From this together with (20) and (21) we obtain $([C]_{c^*} - \boldsymbol{\xi}_{c*}, \boldsymbol{e}_{d^*}) \geq \epsilon/(\alpha_t \beta_t)$. Since the right hand side of this inequality monotonically decreases with respect to t, $([C]_{c^*} - \boldsymbol{\xi}_{c*}, \boldsymbol{e}_{d^*}) \geq \epsilon/(\alpha_N \beta_N)$ holds for all $t \in [0 : N]$. □

Now we come to our main result.

Theorem 1. *For a system Σ and a binary relation $\bar{R}$ satisfying Assumption 1, the algorithm **refine_mesh** terminates in finite iterations. Moreover, its outputs $\mathcal{M}^x$ and $\mathcal{M}^u$ define a quantizer embedding $\mathrm{QE}(\Sigma, Q[\mathcal{M}^x], Q[\mathcal{M}^u])$ that is an N-step abstraction of Σ with respect to $\bar{R}$.*

Proof. **refine_mesh** will not terminate before the error condition (14) is satisfied. On the other hand, from Lemma 4, the size of a cell subdivided at each iteration has a positive lower-bound. Since we subdivide a cell in half, the amount a cell shrinks at each subdivision also has a positive lower-bound. Moreover, the state set X and the input set U are both bounded. These facts prove the theorem.


(a) State mesh for uniform error margin


(c) State mesh for relative error margin

(b) Input mesh for uniform error margin

(d) Input mesh for relative error margin

Fig. 2. Discrete abstraction of a linear system

Table 1. Growth of the number of cells

N	0	1	2	3	4	5	6	7	8	-	20
state grid	64	149	208	238	264	283	293	295	296	-	296
input grid	1	2	4	4	4	4	4	4	4	-	4

(a) Uniform error margin

N	0	1	2	3	4	5	6	7	-	10	11	12	-	20
state grid	4	6	14	42	73	97	109	118	-	120	121	121	-	121
input grid	1	1	1	2	4	6	6	8	-	8	8	10	-	10

(b) Relative error margin

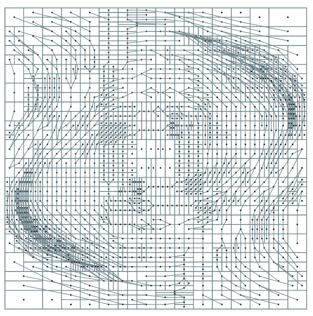
5 Examples

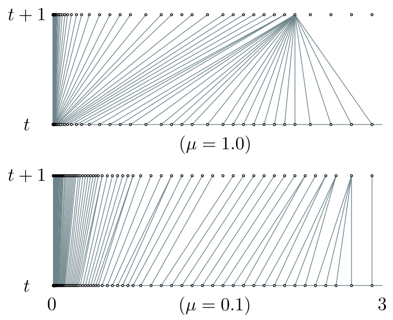
This section shows some simple examples as graphical demonstrations.

Consider the following 2-dimensional linear system:

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

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, at first we compute a discrete abstraction using the uniform error condition $\bar{R} = \{(\boldsymbol{x}, \hat{\boldsymbol{x}}) \mid \|\boldsymbol{x} - \hat{\boldsymbol{x}}\| \leq 0.3\}$. The result is shown in Fig. 2(a),(b). The discrete abstraction obtained is composed of 296 states and 4 inputs. Next, for the same system we specify a relative error condition given as $\bar{R} = \{(\boldsymbol{x}, \hat{\boldsymbol{x}}) \mid \|\boldsymbol{x} - \hat{\boldsymbol{x}}\| \leq 0.2 + 0.7\|\hat{\boldsymbol{x}}\|\}$. This kind of error condition is particularly


(a) Euler stepping



(b) Poincaré section

Fig. 3. Discrete abstraction of val der Pol oscillator.

useful when only the systems behavior near the origin is of interest. The result is shown in Fig. 2(c),(d). In this case, the discrete abstraction is composed of 120 states and 8 inputs. Table 1(a)(b) show how the number of cells grows with respect to the step length N. We can observe that the number of cells does not increase when N becomes larger than a certain value (in these examples, less than 20). This implies a possibility that these discrete abstractions are valid for infinite steps. More theoretical study is needed in a future work.

Next, we consider the discrete abstraction of the van der Pol oscillator as an example of nonlinear periodic systems. The van der Pol oscillator is governed by the following ordinary differential equation:

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

In order to apply our method, this system should be transformed into a discrete-time system. We investigate two ways of time-discretization; Euler stepping and Poincaré section. This time, we choose the Poincaré section as $x \in [0.0, 3.0], \dot{x} = 0$. Fig. 3(a) shows a discrete abstraction of the van der Pol oscillator in the Euler stepping case. The parameters are set as $\mu = 0.5, h = 0.1$ and $N = 6$, where h denotes the step size. The state set is given as $X = [-3, 3] \times [-3, 3]$. The approximation precision is given as $\bar{R} = \{(\boldsymbol{x}, \hat{\boldsymbol{x}}) \mid \|\boldsymbol{x} - \hat{\boldsymbol{x}}\| \leq 0.5\}$. In this result, the discrete abstraction consists of 1294 discrete states. On the other hand, Fig. 3(b) shows the case of Poincaré section. We can observe that the mesh is densely subdivided near the origin. This reflects the fact that a small difference of initial states near the origin could cause a significant difference of the number of cycles required before converging to the periodic orbit.

6 Conclusion

In this paper, we have presented a computational approach to the discrete abstraction of nonlinear systems. The presented approach works well even within the framework of discrete-state abstractions of interconnected systems developed in [15]. That is to say, based on the result of [15], we can treat the case in which our approach is applied to only a complex subsystem of the whole system, which produces a kind of hybrid abstraction.

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. 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.
4. 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.
5. P.E. Caines and Y. Wei : Hierarchical Hybrid Control Systems: A Lattice Theoretic Formulation, IEEE Trans. on Automatic Control, Vol. 43, No. 4, 1998.
6. 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.
7. J. Lunze, B. Nixdorf and J. Schröder : Deterministic Discrete-event Representations of Linear Continuous-variable Systems, Automatica, Vol. 35, 395/406, 1999.
8. 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.
9. 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.
10. A. Girard and G.J. Pappas : Approximation metrics for discrete and continuous systems, IEEE Transactions on Automatic Control, 52(5), 782/798, 2007.
11. A. Girard : Approximately Bisimilar Finite Abstractions of Stable Linear Systems, Hybrid Systems: Computation and Control, vol 4416 in LNCS, 231/244, Springer, 2007.
12. 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.
13. 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.
14. Y. Tazaki, J. Imura; Finite Abstractions of Discrete-time Linear Systems and Its Application to Optimal Control, 17th IFAC World Congress, 2008.
15. Y. Tazaki, J. Imura; Bisimilar Finite Abstractions of Interconnected Systems, Hybrid Systems: Computation and Control, vol 4981 in LNCS, 514/527, Springer, 2008.
16. N. Ramdani, N. Meslem, Y. Candau : Reachability of uncertain nonlinear systems using a nonlinear hybridization, Hybrid Systems: Computation and Control, vol 4981 in LNCS, 415/428, Springer, 2008.