Set Computation for Nonlinear Control

L. Jaulin[1], S. Ratschan[2], L. Hardouin[1]
LISA, université d'Angers, 62 avenue Notre Dame du Lac 49 000 Angers, France.
luc.jaulin@univ-angers.fr
hardouin@istia.univ-angers.fr
[2] Max-Planck-Institut für Informatik, Stuhlsatzenhausenweg 85, 66123 Saarbrücken, Germany
stefan.ratschan@mpi-sb.mpg.de

Keywords: bounded-errors, constraint propagation, CSP, interval analysis, nonlinear control, reliable methods, set computation.

Abstract: This paper proposes a new approach to solve the problem of finding a control sequence for a nonlinear discrete-time system that should satisfy given set-membership specifications on the state and the output vectors. This approach is based on set computation and constraint propagation. Two illustrative examples are provided. The approach is then extended to deal with the robust control problem of nonlinear discrete-time systems.

1 Introduction

Consider the nonlinear discrete-time system described by

$$\left\{ \begin{array}{lcl} \mathbf{x}_k & = & \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}), \\ \mathbf{y}_k & = & \mathbf{g}(\mathbf{x}_k), \end{array} \right. \qquad k \in \{1, \ldots, \bar{k}\}, \tag{1}$$

where $\mathbf{x}_k \in \mathbb{R}^{n_\mathrm{x}}$ is the state vector, $\mathbf{y}_k \in \mathbb{R}^{n_\mathrm{y}}$ is the output vector (which is assumed to be measured) and $\mathbf{u}_k \in \mathbb{R}^{n_\mathrm{u}}$ is the control vector (which can be chosen arbitrarily inside a domain $\mathbb{U}_k$). The initial state vector $\mathbf{x_0}$ is assumed to be known. Functions $\mathbf{f}$ and $\mathbf{g}$ are the evolution and the observation functions at time k. In a set-membership approach, vectors $\mathbf{u}_k$, $\mathbf{x}_k$ and $\mathbf{y}_k$, are assumed to belong to known domains $\mathbb{U}_k$, $\mathbb{X}_k$ and $\mathbb{Y}_k$. Such domains may correspond to prior constraints or specifications on the state, input or output vectors, or to the representation of a measurement. For instance, actuators (providing $\mathbf{u}_k$) are naturally limited in the force (or equivalent) and safety limits state variables, such as temperature, pressure and velocity.

On the one hand, a domain may be arbitrarily large. For instance, if nothing is known about $\mathbf{x}_k$, we shall take $\mathbb{X}_k = \mathbb{R}^{n_\mathrm{x}}$. On the other hand, it may be arbitrarily small. For

instance, if $\mathbf{y}_k$ is measured without any error, $\mathbb{Y}_k$ will be the singleton $\{\widetilde{\mathbf{y}}_k\}$ containing, as single point, the measurement vector $\widetilde{\mathbf{y}}_k$. This paper proposes a new approach based on set computation to solve the classical problem (see [3]) of finding a control sequence $\mathbf{u_0}, \mathbf{u}_1, \ldots, \mathbf{u}_{\bar{k}-1}$ for (1) such that all required set-membership specifications are satisfied (i.e., for all k, $\mathbf{u}_k \in \mathbb{U}_k$, $\mathbf{x}_k \in \mathbb{X}_k$ and $\mathbf{y}_k \in \mathbb{Y}_k$).

The problem, which can be considered as unsolved for general nonlinear systems, is the basic step of *model predictive control* (MPC). MPC (see [9] for a survey) is a form of control in which the current control action is obtained by solving *on line* at each sampling time, the finite horizon open-loop control problem (described above) using the current state as the initial state. The first control vector of the generated sequence is then applied to the system.

Set computation has been widely used for parameter and state estimation [10, 24, 16, 28, 14], in a bounded-error context. It has also been used to deal with robust control and analysis of linear systems [8, 18, 1, 20]. Now, the problem of finding a feasible control sequence $\mathbf{u_0}, \mathbf{u}_1, \ldots, \mathbf{u}_{\bar{k}-1}$ for (1) such that all specifications are satisfied is much more difficult and to the best of our knowledge, it has not been solved yet in the nonlinear case (see [3] for the linear case). This is probably due to the fact that existing set-membership methods are too pessimistic when no bisection is allowed, and limited to very small $\bar{k}$ (typically lower than 3)when bisections are performed [15]. The approach to be proposed in this paper will make it possible to solve our problem even when $\bar{k}$ is large.

In Section 2, the fundamental notion of Constraint Satisfaction Problem (CSP) is presented. The principle of set propagation over a CSP is given in Section 3. The application of set propagation to open and closed loop control of discrete-time systems is presented in Section 4. Two test-cases are given in Section 5. An extension of the approach to the case where uncertainties exist in the system is considered in Section 6.

2 Constraint Satisfaction Problem

2.1 Definitions

A *Constraint Satisfaction Problem* [17] (in short CSP) $\mathcal{H} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ is composed of

1. a set $\mathcal{V}$ of n variables $\mathbf{x}_1, \ldots, \mathbf{x}_n$ (in this section, the $\mathbf{x}_i$'s are vectors of $\mathbb{R}^{n_i}$ and have

nothing to do with the state vector presented in the introduction),

2. a set $\mathcal{D}$ of n subsets $\mathbb{X}_1,\ldots,\mathbb{X}_n$ (called domains) of $\mathbb{R}^{n_i}$ associated with $\mathbf{x}_1,\ldots,\mathbf{x}_n$,

3. a set $\mathcal{C}$ of m binary constraints relating these variables.

A *constraint* $\mathbb{C}_{i,j}$ of $\mathcal{C}$ relating two variables $\mathbf{x}_i$ and $\mathbf{x}_j$ of $\mathcal{V}$ is defined as a subset of $\mathbb{R}^{n_i} \times \mathbb{R}^{n_j}$. The variable $\mathbf{x}_i$ is *consistent* in $\mathcal{H}$ if, for any instantiation $\widetilde{\mathbf{x}}_i$ of $\mathbf{x}_i$ in $\mathbb{X}_i$, it is possible to instantiate all other variables $\mathbf{x}_j$ of $\mathcal{V}$ in their domains such that all constraints of $\mathcal{C}$ are satisfied. The largest subdomain $\hat{\mathbb{X}}_i$ of $\mathbb{X}_i$ that makes $\mathbf{x}_i$ consistent is called the *consistency domain* of $\mathbf{x}_i$. If all domains are equal to their consistency domains, the CSP $\mathcal{H}$ is said to be *globally consistent*. Note that if one of the domains of a globally consistent CSP is empty then all domains of the CSP are empty.

Remark 1 *To our point of view, the name "constraint satisfaction problem", is a bad choice. One may ask what is the problem to be solved. In fact, the problem that is generally considered is to find small enclosures of all consistency domains* $\mathbb{X}_i$.

Two variables $\mathbf{x}_i, \mathbf{x}_j$ are *adjacent* if they are related by a constraint. The associated predicate will be denoted by $\mathrm{adj}(\mathbf{x}_i, \mathbf{x}_j)$, i.e.,

$$\mathrm{adj}(\mathbf{x}_i, \mathbf{x}_j) \Leftrightarrow (\mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are adjacent}) . \tag{2}$$

In what follows, the graph $\mathcal{G}$ of $\mathcal{H}$ (where the nodes and the arcs correspond to the $\mathbf{x}_i$'s and to the constraints, respectively), is assumed to be a tree containing all $\mathbf{x}_i$'s (i.e., with $n-1$ arcs).

Example 1 *Consider a CSP* $\mathcal{H} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ *with 5 variables* $\mathbf{x}_1,\ldots,\mathbf{x}_5$ *and 4 constraints* $\mathbb{C}_{12}$, $\mathbb{C}_{25}$, $\mathbb{C}_{24}$ *and* $\mathbb{C}_{13}$. *Its graph, represented by Figure 1, is a tree. Since* $\mathbf{x}_1$ *and* $\mathbf{x}_2$ *are adjacent, adj(*$\mathbf{x}_1$,$\mathbf{x}_2$*) is true.*

2.2 Consistency relation in a CSP

A *relation* $\mathcal{R}$ of $\mathcal{V}$ is a subset of $\mathcal{V} \times \mathcal{V}$. To indicate that a pair $(\mathbf{x}_i, \mathbf{x}_j)$ belongs to $\mathcal{R}$, we shall use the notation $\mathbf{x}_i \mathcal{R} \mathbf{x}_j$. The relation $\mathcal{R}$ is a *preorder* if it is *reflexive* (i.e., $\forall \mathbf{x}_i \in \mathcal{V}$,

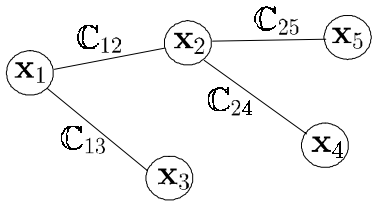

Figure 1: Graph of the CSP $\mathcal{H}$ associated with Example 1

$\mathbf{x}_i \mathcal{R} \mathbf{x}_i$) and *transitive* (i.e., $((\mathbf{x}_i, \mathbf{x}_k) \in \mathcal{R}$ and $(\mathbf{x}_k, \mathbf{x}_j) \in \mathcal{R}) \Rightarrow (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{R}$). The identity relation $\mathcal{I}$ defined by

$$\mathbf{x}_i \mathcal{I} \mathbf{x}_j \Leftrightarrow \mathbf{x}_i = \mathbf{x}_j \tag{3}$$

is the smallest preorder containing $\mathcal{V}$. The *union* $\mathcal{R}_1 \cup \mathcal{R}_2$ of two relations $\mathcal{R}_1$ and $\mathcal{R}_2$ of $\mathcal{V}$ is defined by

$$\mathbf{x}_i \left(\mathcal{R}_1 \cup \mathcal{R}_2\right) \mathbf{x}_j \Leftrightarrow \left(\mathbf{x}_i \mathcal{R}_1 \mathbf{x}_j \text{ or } \mathbf{x}_i \mathcal{R}_2 \mathbf{x}_j\right). \tag{4}$$

The *product* $\mathcal{R}_1.\mathcal{R}_2$ of two relations $\mathcal{R}_1$ and $\mathcal{R}_2$ of $\mathcal{V}$ is defined by

$$\mathbf{x}_i \left(\mathcal{R}_1.\mathcal{R}_2\right) \mathbf{x}_j \Leftrightarrow (\exists \mathbf{x}_k \in \mathcal{V} \mid \mathbf{x}_i \mathcal{R}_1 \mathbf{x}_k \text{ and } \mathbf{x}_k \mathcal{R}_2 \mathbf{x}_j). \tag{5}$$

Consider a relation $\mathcal{R}$ of $\mathcal{V}$ and denote by $\mathcal{R}^*$ the smallest preorder containing $\mathcal{R}$ (i.e., its reflexive/transitive closure). We have

$$\mathcal{R}^* = \mathcal{I} \cup \mathcal{R} \cup \mathcal{R}.\mathcal{R} \cup \mathcal{R}.\mathcal{R}.\mathcal{R} \cup \cdots = \bigcup_{i \geq \mathbf{0}} \mathcal{R}^i \tag{6}$$

where $\mathcal{R}^{\mathbf{0}} = \mathcal{I}$ and $\mathcal{R}^{i+1} = \mathcal{R}.\mathcal{R}^i, i \geq 0$.

Remark 2 *A relation $\mathcal{R}$ can be represented (see e.g. [12]) by a Boolean matrix $\mathbf{R}$ whose entries r_{ij} are equal to 1 if and only if $\mathbf{x}_i \mathcal{R} \mathbf{x}_j$. The matrix counterparts of the union and the product are the addition and the multiplication. For instance, consider the relation $\mathcal{R}$ associated with the matrix*

$$\mathbf{R} \triangleq \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The matrix associated with $\mathcal{R}^*$ is

$$\begin{aligned}
\mathbf{R}^* &= \mathbf{I} + \mathbf{R} + \mathbf{R}^2 + \mathbf{R}^3 \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}$$

Define the relation $\mathcal{G}$ as follows

$$\mathbf{x}_i \mathcal{G} \mathbf{x}_j \Leftrightarrow (\mathrm{adj}(\mathbf{x}_i, \mathbf{x}_j) \text{ and } \forall \widetilde{\mathbf{x}}_i \in \mathbb{X}_i, \exists \widetilde{\mathbf{x}}_j \in \mathbb{X}_j \mid (\widetilde{\mathbf{x}}_i, \widetilde{\mathbf{x}}_j) \in \mathbb{C}_{i,j}). \tag{7}$$

This relation can be interpreted as follows : $\mathbf{x}_i \mathcal{G} \mathbf{x}_j$ if each feasible values for $\mathbf{x}_i$ is consistent with the constraint $\mathbb{C}_{i,j}$ and the domain for $\mathbf{x}_j$.

The smallest preorder relation $\mathcal{G}^*$ which contains $\mathcal{G}$ will be denoted Γ. If $\mathbf{x}_i \Gamma \mathbf{x}_j$, we shall say that $\mathbf{x}_i$ is *consistent* with $\mathbf{x}_j$.

Example 2 *Consider the CSP $\mathcal{H} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ with 5 real variables $x_1, \ldots, x_5$, the graph of which is given by Figure 1. Assume that the constraints are given by*

$$\begin{aligned}
\mathbb{C}_{12} &= \{(\tilde{x}_1, \tilde{x}_2) \mid \tilde{x}_1^2 + \tilde{x}_2^2 \leq 4\}, \\
\mathbb{C}_{25} &= \{(\tilde{x}_2, \tilde{x}_5) \mid \tilde{x}_2 \leq \tilde{x}_5 + 1\}, \\
\mathbb{C}_{24} &= \{(\tilde{x}_2, \tilde{x}_4) \mid \tilde{x}_2 = \exp(\tilde{x}_4 + 3)\}, \\
\mathbb{C}_{13} &= \{(\tilde{x}_1, \tilde{x}_3) \mid \tilde{x}_1 = \sin(\tilde{x}_3)\}.
\end{aligned} \tag{8}$$

and the domains for the variables are given by

$$\mathbb{X}_1 = [0,1],\ \mathbb{X}_2 = [1,10],\ \mathbb{X}_3 = [-5,5],\ \mathbb{X}_4 = [-3,3] \text{ and } \mathbb{X}_5 = [-7,12]. \tag{9}$$

From (7), $x_1 \mathcal{G} x_2$ because $\forall \tilde{x}_1 \in [0,1]$, $\exists \tilde{x}_2 \in [1,10]$ such that $\tilde{x}_1^2 + \tilde{x}_2^2 \leq 4$. With the same manner, we have $x_2 \mathcal{G} x_5, x_2 \mathcal{G} x_4$ and $x_1 \mathcal{G} x_3$ but we do not have $x_2 \mathcal{G} x_1$, $x_5 \mathcal{G} x_2$, $x_4 \mathcal{G} x_2$, and $x_3 \mathcal{G} x_1$.

Proposition 1 (i) If $\mathbf{x}_i$ is consistent with all other variables then $\mathbf{x}_i$ is consistent in $\mathcal{H}$ and its domain $\mathbb{X}_i$ is equal to its consistency domain $\hat{\mathbb{X}}_i$. (ii) if $\mathbf{x}_i$ is consistent with $\mathbf{x}_j$ and if $\mathbb{X}_i$ is replaced by a subset $\mathbb{X}_i'$ of $\mathbb{X}_i$, then $\mathbf{x}_i$ remains consistent with $\mathbf{x}_j$, but the

variables $\mathbf{x}_k, k \neq \mathbf{x}_i$ that were consistent with $\mathbf{x}_i$ may become inconsistent with $\mathbf{x}_i$. (iii) All variables are consistent with all other variables if and only if all domains of $\mathcal{H}$ are equal to their consistency domain (i.e., $\mathcal{H}$ is globally consistent). These propositions can be written in a mathematical form as follows:

$$\begin{array}{ll} \text{(i)} & (\forall \mathbf{x}_j \in \mathcal{V}, \mathbf{x}_i \Gamma \mathbf{x}_j) \Rightarrow \mathbb{X}_i = \hat{\mathbb{X}}_i, \\ \text{(ii)} & \text{if } \mathbf{x}_i \Gamma \mathbf{x}_j \text{ and if } \mathbb{X}_i \text{ is replaced by a set } \mathbb{X}_i' \subset \mathbb{X}_i \text{ then we still have } \mathbf{x}_i \Gamma \mathbf{x}_j, \\ \text{(iii)} & (\forall \mathbf{x}_i \in \mathcal{V}, \forall \mathbf{x}_j \in \mathcal{V}, \mathbf{x}_i \Gamma \mathbf{x}_j) \Leftrightarrow (\forall \mathbf{x}_i \in \mathcal{V}, \mathbb{X}_i = \hat{\mathbb{X}}_i). \end{array} \tag{10}$$

Proof: We shall only give a sketch of the proof of (i) on an example based on the CSP represented in Figure 1. Assume that $(\forall \mathbf{x}_j \in \mathcal{V}, \mathbf{x}_1 \Gamma \mathbf{x}_j)$ and let us show that $\mathbf{x}_1$ is consistent. We have $\mathbf{x}_1 \Gamma \mathbf{x}_2$, $\mathbf{x}_1 \Gamma \mathbf{x}_3$, $\mathbf{x}_2 \Gamma \mathbf{x}_4$ and $\mathbf{x}_2 \Gamma \mathbf{x}_5$. Now, from (7),

$$\begin{array}{lrcl} \text{(a)} & \mathbf{x}_1 \Gamma \mathbf{x}_2 & \Rightarrow & \forall \widetilde{\mathbf{x}}_1 \in \mathbb{X}_1, \exists \widetilde{\mathbf{x}}_2 \in \mathbb{X}_2 \mid (\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2) \in \mathbb{C}_{12}, \\ \text{(b)} & \mathbf{x}_2 \Gamma \mathbf{x}_5 & \Rightarrow & \forall \widetilde{\mathbf{x}}_2 \in \mathbb{X}_2, \exists \widetilde{\mathbf{x}}_5 \in \mathbb{X}_5 \mid (\widetilde{\mathbf{x}}_2, \widetilde{\mathbf{x}}_5) \in \mathbb{C}_{25}, \\ \text{(c)} & \mathbf{x}_2 \Gamma \mathbf{x}_4 & \Rightarrow & \forall \widetilde{\mathbf{x}}_2 \in \mathbb{X}_2, \exists \widetilde{\mathbf{x}}_4 \in \mathbb{X}_4 \mid (\widetilde{\mathbf{x}}_2, \widetilde{\mathbf{x}}_4) \in \mathbb{C}_{24}, \\ \text{(d)} & \mathbf{x}_1 \Gamma \mathbf{x}_3 & \Rightarrow & \forall \widetilde{\mathbf{x}}_1 \in \mathbb{X}_1, \exists \widetilde{\mathbf{x}}_3 \in \mathbb{X}_3 \mid (\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_3) \in \mathbb{C}_{13}. \end{array} \tag{11}$$

From (a) and (d), we get:

$$\forall \widetilde{\mathbf{x}}_1 \in \mathbb{X}_1, \exists \widetilde{\mathbf{x}}_2 \exists \widetilde{\mathbf{x}}_3 \begin{pmatrix} \widetilde{\mathbf{x}}_2 \in \mathbb{X}_2 \\ \widetilde{\mathbf{x}}_3 \in \mathbb{X}_3 \end{pmatrix} \text{ such that } \begin{pmatrix} (\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2) \in \mathbb{C}_{12} \\ (\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_3) \in \mathbb{C}_{13} \end{pmatrix} \tag{12}$$

From (b),

$$\forall \widetilde{\mathbf{x}}_1 \in \mathbb{X}_1, \exists \widetilde{\mathbf{x}}_2 \exists \widetilde{\mathbf{x}}_3 \exists \widetilde{\mathbf{x}}_5 \begin{pmatrix} \widetilde{\mathbf{x}}_2 \in \mathbb{X}_2 \\ \widetilde{\mathbf{x}}_3 \in \mathbb{X}_3 \\ \widetilde{\mathbf{x}}_5 \in \mathbb{X}_5 \end{pmatrix} \text{ such that } \begin{pmatrix} (\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2) \in \mathbb{C}_{12} \\ (\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_3) \in \mathbb{C}_{13} \\ (\widetilde{\mathbf{x}}_2, \widetilde{\mathbf{x}}_5) \in \mathbb{C}_{25} \end{pmatrix} \tag{13}$$

and from (c)

$$\forall \widetilde{\mathbf{x}}_1 \in \mathbb{X}_1, \exists \widetilde{\mathbf{x}}_2 \exists \widetilde{\mathbf{x}}_3 \exists \widetilde{\mathbf{x}}_4 \exists \widetilde{\mathbf{x}}_5 \begin{pmatrix} \widetilde{\mathbf{x}}_2 \in \mathbb{X}_2 \\ \widetilde{\mathbf{x}}_3 \in \mathbb{X}_3 \\ \widetilde{\mathbf{x}}_4 \in \mathbb{X}_4 \\ \widetilde{\mathbf{x}}_5 \in \mathbb{X}_5 \end{pmatrix} \text{ such that } \begin{pmatrix} (\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2) \in \mathbb{C}_{12} \\ (\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_3) \in \mathbb{C}_{13} \\ (\widetilde{\mathbf{x}}_2, \widetilde{\mathbf{x}}_5) \in \mathbb{C}_{25} \\ (\widetilde{\mathbf{x}}_2, \widetilde{\mathbf{x}}_4) \in \mathbb{C}_{24} \end{pmatrix}. \tag{14}$$

Therefore $\mathbf{x}_1$ is consistent in $\mathcal{H}$. ■

2.3 Contraction operator

If $\text{adj}(\mathbf{x}_i, \mathbf{x}_j)$ is true, define the contraction operator $\rho_{\mathbf{x}_i}^{\mathbf{x}_j}$ as follows

$$\rho_{\mathbf{x}_i}^{\mathbf{x}_j}(\mathbb{X}_i) \triangleq \{\widetilde{\mathbf{x}}_i \in \mathbb{X}_i \mid \exists \widetilde{\mathbf{x}}_j \in \mathbb{X}_j,\ (\widetilde{\mathbf{x}}_i, \widetilde{\mathbf{x}}_j) \in \mathbb{C}_{ij}\}.$$

Note that $\rho_{\mathbf{x}_i}^{\mathbf{x}_j}(\mathbb{X}_i)$ is the largest subdomain of $\mathbb{X}_i$ that can replace $\mathbb{X}_i$ in order to have $\mathbf{x}_i \Gamma \mathbf{x}_j$.

Example 3 *Consider the CSP treated in Example 2. We have*

$$\begin{array}{rcll} \rho_{x_2}^{x_1}(\mathbb{X}_2) & = & \{\tilde{x}_2 \in [0,10] \mid \exists \tilde{x}_1 \in [0,1],\ \tilde{x}_1^2 + \tilde{x}_2^2 \leq 4\} & = [0,2], \\ \rho_{x_1}^{x_2}(\mathbb{X}_1) & = & \{\tilde{x}_1 \in [0,1] \mid \exists \tilde{x}_2 \in [0,10],\ \tilde{x}_1^2 + \tilde{x}_2^2 \leq 4\} & = [0,1], \\ \rho_{x_5}^{x_2}(\mathbb{X}_5) & = & \{\tilde{x}_5 \in [-7,12] \mid \exists \tilde{x}_2 \in [0,10],\ \tilde{x}_2 \leq \tilde{x}_5 + 1\} & = [1,12]. \end{array} \quad (15)$$

Interval computation [22] may help to obtain these intervals. For instance,

$$\begin{array}{rcl} \rho_{x_5}^{x_2}(\mathbb{X}_5) & = & \{\tilde{x}_5 \in [-7,12] \mid \exists \tilde{x}_2 \in [0,10], \exists a \in [0,\infty[,\ \tilde{x}_5 = \tilde{x}_2 + a - 1\} \\ & = & [-7,12] \cap ([0,10] + [0,\infty[+1) = [-7,12] \cap [1,\infty] = [1,12]. \end{array} \quad (16)$$

Example 4 *Assume that the constraint between the two adjacent variables $\mathbf{x}_i$ and $\mathbf{x}_j$ is given $\mathbb{C}_{i,j} = \{(\widetilde{\mathbf{x}}_i, \widetilde{\mathbf{x}}_j) \in \mathbb{R}^{n_i} \times \mathbb{R}^{n_j} \mid \widetilde{\mathbf{x}}_i = \mathbf{f}(\widetilde{\mathbf{x}}_j)\}$. Then,*

$$\begin{array}{rcl} \rho_{\mathbf{x}_i}^{\mathbf{x}_j}(\mathbb{X}_i) & = & \{\widetilde{\mathbf{x}}_i \in \mathbb{X}_i \mid \exists \widetilde{\mathbf{x}}_j \in \mathbb{X}_j,\ \widetilde{\mathbf{x}}_i = \mathbf{f}(\widetilde{\mathbf{x}}_j)\} = \mathbf{f}(\mathbb{X}_j) \cap \mathbb{X}_i, \\ \rho_{\mathbf{x}_j}^{\mathbf{x}_i}(\mathbb{X}_j) & = & \{\widetilde{\mathbf{x}}_j \in \mathbb{X}_j \mid \exists \widetilde{\mathbf{x}}_i \in \mathbb{X}_i,\ \widetilde{\mathbf{x}}_i = \mathbf{f}(\widetilde{\mathbf{x}}_j)\} = \mathbf{f}^{-1}(\mathbb{X}_i) \cap \mathbb{X}_j. \end{array} \quad (17)$$

Proposition 2: In $\mathcal{H}$, if $\mathbf{x}_i$ and $\mathbf{x}_j$ are adjacent, the statement $\mathbb{X}_i := \rho_{\mathbf{x}_i}^{\mathbf{x}_j}(\mathbb{X}_i)$ leaves the consistency domains of $\mathcal{H}$ unchanged. Moreover, after this statement, $\mathbf{x}_i \Gamma \mathbf{x}_j$.

Proof: The consistency domain $\hat{\mathbb{X}}_k$ of $\mathbf{x}_k$ is made with all $\widetilde{\mathbf{x}}_k$'s of $\mathbb{X}_k$ such that all other variables can be instantiated in their domains so that all constraints are satisfied. Now, the statement $\mathbb{X}_i := \rho_{\mathbf{x}_i}^{\mathbf{x}_j}(\mathbb{X}_i)$ eliminates values for $\mathbf{x}_i$ that cannot satisfy the constraint $\mathbb{C}_{ij}$ and therefore, this statement cannot modify $\hat{\mathbb{X}}_k$. The fact that the statement $\mathbb{X}_i := \rho_{\mathbf{x}_i}^{\mathbf{x}_j}(\mathbb{X}_i)$ yields the property $\mathbf{x}_i \Gamma \mathbf{x}_j$ is a direct consequence of the definition of Γ (see equation (7)). ■

3 Set propagation

Set propagation will make it possible to compute consistency domains. The ideas presented here are taken from [5], [21] and [2].

Notation: Recall that the graph of the CSP $\mathcal{H}$ has been assumed to be a tree $\mathcal{G}$. Let $\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_k}$ be k variables of $\mathcal{H}$. The smallest subtree of $\mathcal{G}$ which contains $\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_k}$ will

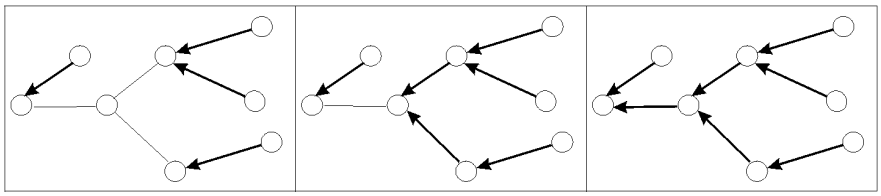

Figure 2: Sequencing of the contractions in FALL; the root $\mathbf{x}_i$ of the tree $\mathcal{T}$ corresponds to the leftmost node;

be denoted by $\mathcal{T}(\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_k})$. The set of all $\mathbf{x}_i$ involved in a given subtree $\mathcal{T}$ of $\mathcal{G}$ will be denoted by $\mathrm{var}(\mathcal{T})$. Note that $\mathrm{var}(\mathcal{T}(\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_k})) \supset \{\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_k}\}$.

Let $\mathcal{T}$ be a subtree of $\mathcal{G}$, such that $\mathbf{x}_i \in \mathrm{var}(\mathcal{T})$. Consider the recursive following algorithm.

Algorithm FALL$(\mathbf{x}_i, \mathcal{T})$	
1	mark $\mathbf{x}_i$;
2	for all $\mathbf{x}_j \in \mathcal{V}$ such that $\mathrm{adj}(\mathbf{x}_i, \mathbf{x}_j)$;
3	if $\mathbf{x}_j$ is not marked and if $\mathbf{x}_j$ is a variable of $\mathcal{T}$;
4	FALL$(\mathbf{x}_j, \mathcal{T})$;
5	$\mathbb{X}_i := \rho_{\mathbf{x}_i}^{\mathbf{x}_j}(\mathbb{X}_i)$;
6	end if;
5	end for.

(18)

The name FALL is due to the fact that $\mathbf{x}_i$ can be considered as the root of $\mathcal{T}$ and that algorithm operates from the leaves of $\mathcal{T}$ down to its root. As for botanical trees, the root is at the bottom of the tree, and the leaves at the top.

Proposition 3: After completion of FALL$(\mathbf{x}_i, \mathcal{T})$, $\mathbf{x}_i$ is consistent with all variables of $\mathrm{var}(\mathcal{T})$ (i.e., $\forall \mathbf{x}_j \in \mathrm{var}(\mathcal{T}), \mathbf{x}_i \Gamma \mathbf{x}_j$).

This proposition, proven in [13], is illustrated by the consistency graph of Figure 2. This graph represents the sequencing of the operations of FALL. An arrow from the node $\mathbf{x}_j$ toward the node $\mathbf{x}_i$ indicates that the operation $\mathbb{X}_i := \rho_{\mathbf{x}_i}^{\mathbf{x}_j}(\mathbb{X}_i)$ has been performed which implies that the relation $\mathbf{x}_i \Gamma \mathbf{x}_j$ holds true.

Consider the recursive following algorithm where all variables have initially been un-

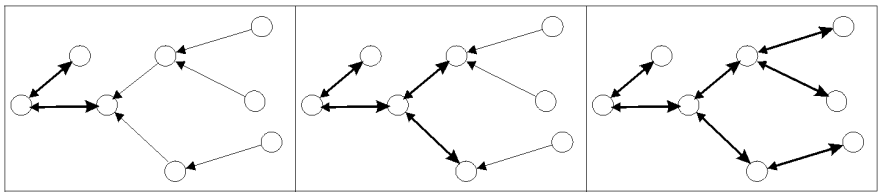

Figure 3: Sequencing of the operations in CLIMB; the contractions are performed from the root upto its leaves

marked.

Algorithm CLIMB$(\mathbf{x}_i, \mathcal{T})$	
1	mark $\mathbf{x}_i$;
2	for all $\mathbf{x}_j \in \mathcal{V}$ such that adj$(\mathbf{x}_i, \mathbf{x}_j)$;
3	if $\mathbf{x}_j$ is not marked and if $\mathbf{x}_j$ is a variable of $\mathcal{T}$
4	$\mathbb{X}_j := \rho_{\mathbf{x}_j}^{\mathbf{x}_i}(\mathbb{X}_j)$;
5	CLIMB$(\mathbf{x}_j, \mathcal{T})$;
6	end if;
5	end for.

The algorithm operates from the root $\mathbf{x}_i$ of $\mathcal{T}$ upto its leaves.

Proposition 4 (proven in [13]): If $(\forall \mathbf{x}_j \in \text{var}(\mathcal{T}), \mathbf{x}_i \Gamma \mathbf{x}_j)$, after completion of CLIMB$(\mathbf{x}_i, \mathcal{T})$, we have $(\forall \mathbf{x}_j \in \text{var}(\mathcal{T}), \forall \mathbf{x}_k \in \text{var}(\mathcal{T}), \mathbf{x}_j \Gamma \mathbf{x}_k)$.

This proposition is illustrated by the consistency graphs of Figure 3 which depicts the sequencing of the operations in CLIMB. Small arrows indicate that before running CLIMB, the root (leftmost node) $\mathbf{x}_i$ is consistent with all other nodes. Big arrows indicate that the operation $\mathbb{X}_j := \rho_{\mathbf{x}_j}^{\mathbf{x}_i}(\mathbb{X}_j)$ has been performed by CLIMB and thus that the relation $\mathbf{x}_j \Gamma \mathbf{x}_i$ is valid. After completion of CLIMB, if the subtree $\mathcal{T}$ corresponds to the whole graph $\mathcal{G}$ of the CSP $\mathcal{H}$, then $\mathcal{H}$ is globally consistent.

Example 5 *Consider the CSP $\mathcal{H}$ with 5 variables $\mathbf{x}_1, \ldots, \mathbf{x}_5$ related by the following constraints*

$$\begin{cases} \mathbf{x}_1 = \mathbf{f}_3(\mathbf{x}_3), \\ \mathbf{x}_4 = \mathbf{f}_4(\mathbf{x}_2), \\ \mathbf{x}_2 = \mathbf{f}_5(\mathbf{x}_5), \\ \mathbf{x}_2 = \mathbf{f}_2(\mathbf{x}_1). \end{cases} \tag{19}$$

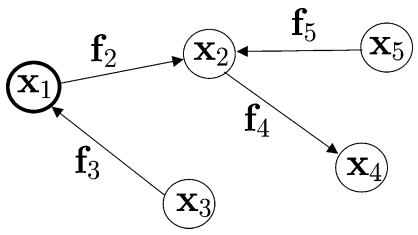

Figure 4: The graph $\mathcal{G}$ of the CSP is a tree, the chosen root is $\mathbf{x}_1$

The graph $\mathcal{G}$ of $\mathcal{H}$ is given by Figure 4, where arrows indicate the direction in which the functions apply. Since it does not contain any cycle, this graph is a tree. The tree $\mathcal{T}(\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5)$ is equal to the initial tree $\mathcal{G}$. The tree $\mathcal{T}(\mathbf{x}_1, \mathbf{x}_4)$ is a chain with the 3 nodes $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_4$. The consistency domains $\hat{\mathbb{X}}_1$ of $\mathbf{x}_1$ is obtained by FALL$(\mathbf{x}_1, \mathcal{G})$*. In this context,* FALL *translates into the following algorithm*

Algorithm FALL(*inout:* $\mathbb{X}_1, \mathbb{X}_2, \mathbb{X}_3, \mathbb{X}_4, \mathbb{X}_5$)		
1	$\mathbb{X}_2 := \mathbb{X}_2 \cap \mathbf{f}_5(\mathbb{X}_5)$;	// $\mathbb{X}_2 := \rho_{\mathbf{x}_2}^{\mathbf{x}_5}(\mathbb{X}_2)$
2	$\mathbb{X}_2 := \mathbb{X}_2 \cap \mathbf{f}_4^{-1}(\mathbb{X}_4)$;	// $\mathbb{X}_2 := \rho_{\mathbf{x}_2}^{\mathbf{x}_4}(\mathbb{X}_2)$
3	$\mathbb{X}_1 := \mathbb{X}_1 \cap \mathbf{f}_3(\mathbb{X}_3)$;	// $\mathbb{X}_1 := \rho_{\mathbf{x}_1}^{\mathbf{x}_3}(\mathbb{X}_1)$
4	$\mathbb{X}_1 := \mathbb{X}_1 \cap \mathbf{f}_2^{-1}(\mathbb{X}_2)$;	// $\mathbb{X}_1 := \rho_{\mathbf{x}_1}^{\mathbf{x}_2}(\mathbb{X}_1)$

(20)

After Step 1, $\mathbf{x}_2 \Gamma \mathbf{x}_5$; after Step 2, $\mathbf{x}_2 \Gamma \mathbf{x}_4$; after Step 3, $\mathbf{x}_1 \Gamma \mathbf{x}_3$; after Step 4, $\mathbf{x}_1 \Gamma \mathbf{x}_2$. Thus $\mathbf{x}_1$ is consistent with $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$, $\mathbf{x}_5$ and $\mathbb{X}_1$ is equal to its consistency domain $\hat{\mathbb{X}}_1$.

Example 6 *Consider the CSP of Example 5 and assume that $\forall\ \mathbf{x}_j \in \mathcal{V}$, $\mathbf{x}_1 \Gamma \mathbf{x}_j$. To get the consistency domain of a given variable, say $\mathbf{x}_5$, it suffices to call* FALL$(\mathbf{x}_5, \mathcal{T}(\mathbf{x}_1, \mathbf{x}_5))$*, which translates into the sequence*

$$\mathbb{X}_2 := \mathbb{X}_2 \cap \mathbf{f}_2(\mathbb{X}_1)\,;\ \mathbb{X}_5 := \mathbb{X}_5 \cap \mathbf{f}_5^{-1}(\mathbb{X}_2)\,; \tag{21}$$

After completion of the sequence, $\mathbf{x}_5 \Gamma \mathbf{x}_1$. By transitivity, $\forall\ \mathbf{x}_j \in \mathcal{G}$, $\mathbf{x}_5 \Gamma \mathbf{x}_j$.

Example 7 *Consider again the CSP of Examples 5 and 6 and assume that $\forall\ \mathbf{x}_j \in \mathcal{G}$, $\mathbf{x}_1 \Gamma \mathbf{x}_j$. To compute all consistency domains, it suffices to run* CLIMB$(\mathbf{x}_1, \mathcal{G})$*. In this*

context, it translates into the following algorithm.

Algorithm CLIMB(*inout:* $\mathbb{X}_1, \mathbb{X}_2, \mathbb{X}_3, \mathbb{X}_4, \mathbb{X}_5$)		
5	$\mathbb{X}_2 := \mathbb{X}_2 \cap \mathbf{f}_2(\mathbb{X}_1)$;	// $\mathbb{X}_2 := \rho^{\mathbf{x}_1}_{\mathbf{x}_2}(\mathbb{X}_2)$
6	$\mathbb{X}_3 := \mathbb{X}_3 \cap \mathbf{f}_3^{-1}(\mathbb{X}_1)$;	// $\mathbb{X}_3 := \rho^{\mathbf{x}_1}_{\mathbf{x}_3}(\mathbb{X}_3)$
7	$\mathbb{X}_5 := \mathbb{X}_5 \cap \mathbf{f}_5^{-1}(\mathbb{X}_2)$;	// $\mathbb{X}_5 := \rho^{\mathbf{x}_2}_{\mathbf{x}_5}(\mathbb{X}_5)$
8	$\mathbb{X}_4 := \mathbb{X}_4 \cap \mathbf{f}_4(\mathbb{X}_2)$;	// $\mathbb{X}_4 := \rho^{\mathbf{x}_2}_{\mathbf{x}_4}(\mathbb{X}_4)$

(22)

After completion of CLIMB $\mathcal{H}$ *is globally consistent.*

Remark 3 *When the graph is not a tree and when the constraints are not binary, it is possible to group variables into vectors to build a new CSP, the graph of which is a tree. This is the purpose of so-called clustering techniques [7].*

The computation of $\mathbf{f}(\mathbb{X})$, $\mathbf{f}^{-1}(\mathbb{Y})$ or $\mathbb{X} \cap \mathbb{Y}$, needed by FALL and CLIMB, can only be approximated in the general case [13]. However in some particular cases it is possible to perform these tasks exactly and in a finite number of steps. This is illustrated by the following examples.

Example 8 *When all $\mathbb{X}_i$'s are polytopes and when the constraints $\mathbf{x}_j = \mathbf{f}(\mathbf{x}_i)$ are such that $\mathbf{f}$ is linear, all domains handled by* FALL *and* CLIMB *will also be polytopes [27]. The computation of the consistency domains of the CSP can thus be done in a finite and exact way.*

Example 9 *Assume that all functions $\mathbf{f}$ involved in the CSP are such that for all boxes $[\mathbf{x}]$ and $[\mathbf{y}]$ of appropriate dimensions, $\mathbf{f}([\mathbf{x}])$ and $\mathbf{f}^{-1}([\mathbf{y}])$ are boxes. If all $\mathbb{X}_i$'s are boxes then all consistency domains are also boxes and can be computed exactly by* FALL *and* CLIMB*. An example of such a box-conservative function is*

$$\mathbf{f}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -x_2^3 + 2 \\ x_1 + \exp(x_1) \end{pmatrix}. \tag{23}$$

Example 10 *A* corner *is a subset of $\mathbb{R}^n$ of the form $[\mathbf{x}] = [x_1^-, \infty[\times \cdots \times [x_n^-, \infty[$. There exists some functions that are corner-conservative. For instance, the inverse of the corner $\mathbb{Y} = [y_1^-, \infty[\times [y_2^-, \infty[$ by the function*

$$\mathbf{f}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \min(x_1 + 2, 3x_2 + 4) \\ \min(5x_1 + 6, 7x_2 + 8) \end{pmatrix}. \tag{24}$$

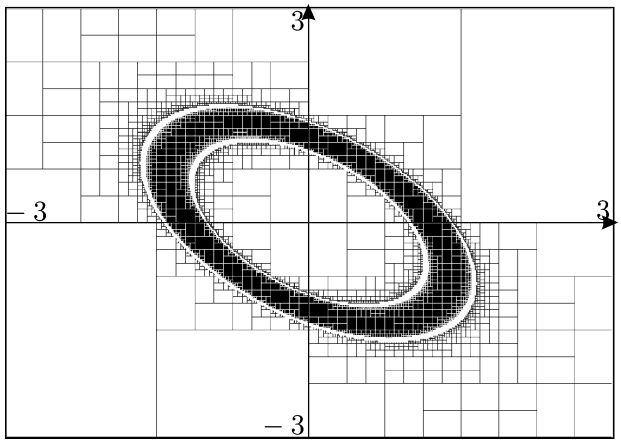

Figure 5: Inner and outer approximations of $\mathbb{X}$

is the corner

$$\mathbf{f}^{-1}\left(\begin{array}{c}[y_1^-,\infty[\\ [y_2^-,\infty[\end{array}\right)=\left(\begin{array}{c}[\max\left(y_1^- - 2, \frac{y_2^- - 6}{5}\right),\infty[\\ [\max\left(\frac{y_1^- - 4}{3}, \frac{y_2^- - 8}{7}\right),\infty[\end{array}\right). \tag{25}$$

If all domains are corners, the consistency domains can be computed exactly provided that only corner-conservative functions are involved in the set propagation.

Example 11 *General compact sets of $\mathbb{R}^n$ can be approximated by union of boxes with an arbitrary accuracy. For instance, an inner and outer approximation of the set $\mathbb{X} \triangleq \{(x_1,x_2) \mid 1 \leq x_1^2 + x_2^2 + x_1x_2 \leq 2\}$ is depicted in Figure 5. Interval methods can thus be used to compute the reciprocal or the direct images of $\mathbb{X}$ by nonlinear function in a guaranteed way [14]. For instance, the reciprocal image of $\mathbb{X}$ by the function*

$$\mathbf{f}: \left(\begin{array}{c}x_1\\ x_2\end{array}\right) \rightarrow \left(\begin{array}{c}\sin(x_1)+x_2^2\\ x_1^2+\sin(x_2)\end{array}\right)$$

is approximated as illustrated by Figure 6. This approximation has been obtained by the algorithm SIVIA *(Set Inversion Via Interval Analysis [14]) in 0.5 sec. on a Pentium 300.*

Remark 4 *If we consider the set operation 'computing $\rho^{\mathbf{x}_i}_{\mathbf{x}_j}(\mathbb{X}_j)$ exactly' as an elementary operation, then the complexity of both algorithms* FALL *and* CLIMB *is linear with respect to the number of variables in the tree. Unfortunately, often in practice, $\rho^{\mathbf{x}_i}_{\mathbf{x}_j}(\mathbb{X}_j)$ can only be approximated and computing its approximation can only be performed by an algorithm whose complexity is exponential with respect to the dimension of $\mathbb{X}_j$.*

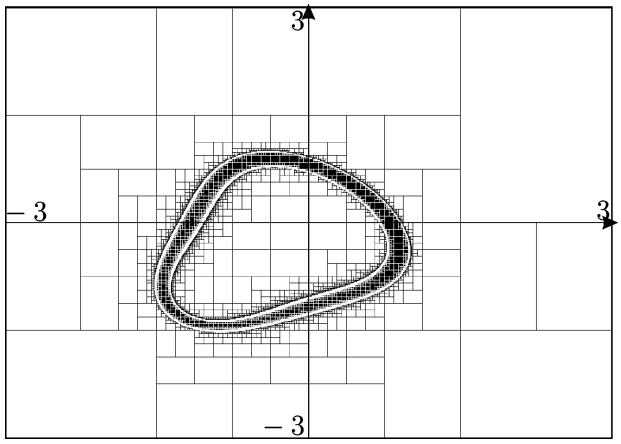

Figure 6: The reciprocal image of $\mathbb{X}$ by $\mathbf{f}$ computed by the algorithm SIVIA

4 Application to control

State-space equations (1) can be cast into binary relations as follows:

$$\left\{\begin{array}{lcl}\mathbf{x}_k &=& \mathbf{f}(\mathbf{x}_{k-1},\mathbf{u}_{k-1}),\\ \mathbf{y}_k &=& \mathbf{g}(\mathbf{x}_k),\end{array}\right. \Leftrightarrow \left\{\begin{array}{lcl}\mathbf{x}_k &=& \mathbf{f}(\mathbf{z}_{k-1}),\\ \mathbf{x}_k &=& \pi_{\mathbf{x}}(\mathbf{z}_k),\\ \mathbf{u}_k &=& \pi_{\mathbf{u}}(\mathbf{z}_k),\\ \mathbf{y}_k &=& \mathbf{g}(\mathbf{x}_k).\end{array}\right. . \tag{26}$$

where $k \in \{1,\ldots,\bar{k}\}$ and where $\pi_{\mathbf{x}}$ and $\pi_{\mathbf{u}}$ are the projection operators defined by the following equivalence

$$\mathbf{z} = (\mathbf{x}\ \ \mathbf{u}) \Leftrightarrow (\mathbf{u} = \pi_{\mathbf{u}}(\mathbf{z}) \text{ and } \mathbf{x} = \pi_{\mathbf{x}}(\mathbf{z}))\,. \tag{27}$$

Our control problem can be represented by the CSP $\mathcal{H} = (\mathcal{V},\mathcal{D},\mathcal{C})$ with

$$\begin{array}{lcl}\mathcal{V} &=& \{\mathbf{u}_0,\ldots,\mathbf{u}_{\bar{k}-1},\mathbf{x}_0,\mathbf{x}_1,\ldots,\mathbf{x}_{\bar{k}},\mathbf{z}_0,\ldots,\mathbf{z}_{\bar{k}},\mathbf{y}_1,\ldots,\mathbf{y}_{\bar{k}}\}\,,\\ \mathcal{D} &=& \{\mathbb{U}_0,\ldots,\mathbb{U}_{\bar{k}-1},\mathbb{X}_0,\mathbb{X}_1,\ldots,\mathbb{X}_{\bar{k}},\mathbb{R}^{n_z},\ldots,\mathbb{R}^{n_z},\mathbb{Y}_1,\ldots,\mathbb{Y}_{\bar{k}}\}\,,\\ \mathcal{C} &=& \{\mathbf{x}_k = \mathbf{f}(\mathbf{z}_{k-1}),\mathbf{x}_k = \pi_{\mathbf{x}}(\mathbf{z}_k),\mathbf{u}_k = \pi_{\mathbf{u}}(\mathbf{z}_k),\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k),k = 1,\ldots,\bar{k}\}\,.\end{array} \tag{28}$$

The graph $\mathcal{G}$ of $\mathcal{H}$ is the tree shown in Figure 7.

4.1 Open-loop set controller

Let us recall that $\mathbf{x}_0$ is known, *i.e.*, its domain $\mathbb{X}_0$ is a singleton $\{\widetilde{\mathbf{x}}_0\}$. The control sequence $(\widetilde{\mathbf{u}}_0,\ldots,\widetilde{\mathbf{u}}_{\bar{k}-1}) \in \mathbb{U}_0 \times \cdots \times \mathbb{U}_{\bar{k}-1}$ is *feasible* if the simulation of (1) leads to

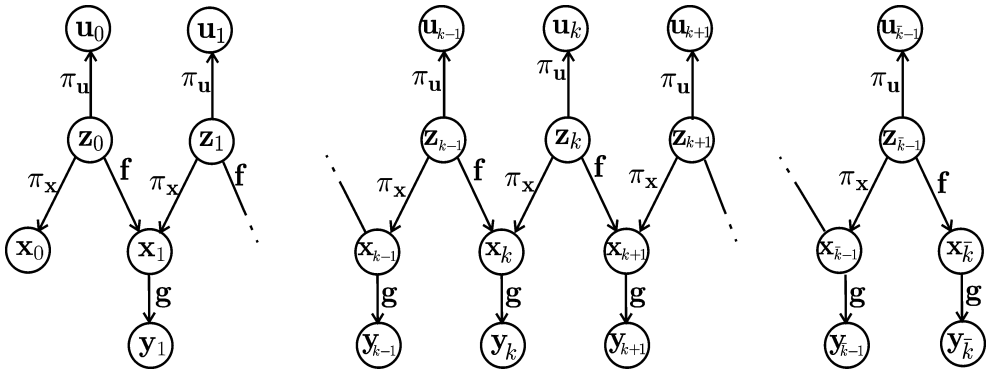

Figure 7: Graph $\mathcal{G}$ associated with the system to be controlled

$\mathbf{x}_k$'s and $\mathbf{y}_k$'s that belong to their domains. The following algorithm computes a feasible sequence $\widetilde{\mathbf{u}}_0, \ldots, \widetilde{\mathbf{u}}_{\bar{k}-1}$.

Algorithm OpenLoopControl(in: $\mathbb{X}_0, \mathbb{U}_0, \mathbb{Y}_1, \ldots, \mathbb{X}_{\bar{k}}, \mathbb{U}_{\bar{k}-1}, \mathbb{Y}_{\bar{k}}$; out $\widetilde{\mathbf{u}}_0, \ldots, \widetilde{\mathbf{u}}_{\bar{k}-1}$)	
1	$\mathbb{Z}_0 := \mathbb{R}^{n_z}$; ...; $\mathbb{Z}_{\bar{k}-1} := \mathbb{R}^{n_z}$;
2	Fall($\mathbf{u}_0$,$\mathcal{G}$);
3	if $\mathbb{U}_0 = \emptyset$, return ("no control can be found");
4	for $k := 1$ to $\bar{k} - 1$
5	choose $\widetilde{\mathbf{u}}_{k-1} \in \mathbb{U}_{k-1}$; $\mathbb{U}_{k-1} := \{\widetilde{\mathbf{u}}_{k-1}\}$;
6	Fall($\mathbf{u}_k, \mathcal{T}(\mathbf{u}_{k-1}, \mathbf{u}_k)$);
7	choose $\widetilde{\mathbf{u}}_{\bar{k}-1} \in \mathbb{U}_{\bar{k}-1}$.

Remark 5 *If we consider set operations such as (i) computing the direct image, (ii) computing the reciprocal image, (iii) intersecting, (iv) projecting, ... as elementary operations, then the complexity of* OpenLoopControl *is linear with respect to the number $\bar{k}$. Again, recall, that, in practice, these operations can only be approximated and their complexities are often exponential with respect to n_z.*

Comments: Step 2 computes the consistency domain $\widehat{\mathbb{U}}_0$ of $\mathbf{u}_0$. After Step 2, $\mathbf{u}_0$ is consistent with all other variables, i.e., if the relation $\mathbf{x}_i \Gamma \mathbf{x}_j$ is represented by an arrow from $\mathbf{x}_j$ to $\mathbf{x}_i$, there exists a path from any other variable to $\mathbf{u}_0$ (see the consistency graph of Figure 8 (a)). If $\widehat{\mathbb{U}}_0$ is empty, no feasible control sequence can be found and the algorithm stop at Step 3. At Step 5, one $\widetilde{\mathbf{u}}_0$ is chosen in $\widehat{\mathbb{U}}_0$. After the statement $\mathbb{U}_0 := \{\widetilde{\mathbf{u}}_0\}$, $\mathbf{u}_0$ will remain consistent with all other variables (see (ii) of Proposition 1) and the associated oriented tree is still given by Figure 8 (a). The consistency domain

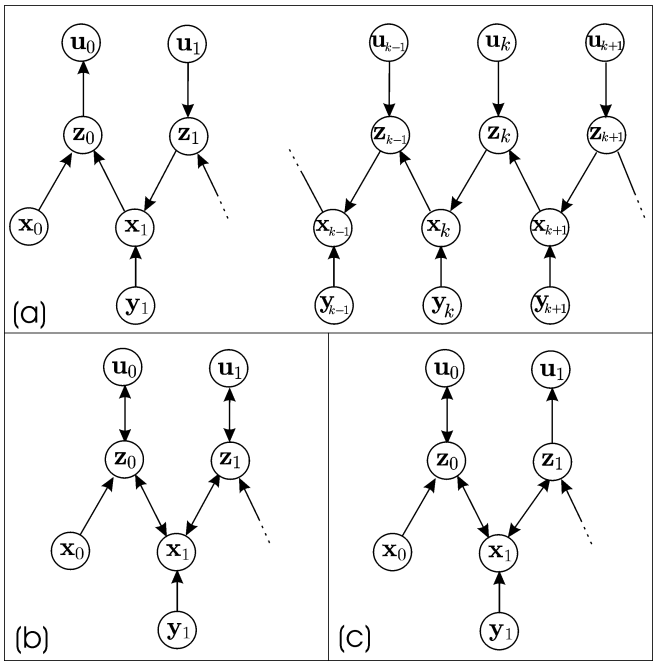

Figure 8: Principle of the algorithm OpenLoopControl

for $\mathbf{u}_1$ is then obtained by Fall$(\mathbf{u}_1, \mathcal{T}(\mathbf{u_0}, \mathbf{u}_1))$ at Step 6 (see Example 6). After Step 6, the left part of consistency graph is given by Figure 8 (b). At this stage, both $\mathbf{u_0}$ and $\mathbf{u}_1$ are consistent with all other nodes. After instantiation of $\mathbf{u}_1$, the variable $\mathbf{u}_1$ remains consistent with all other variables, but $\mathbf{z}_1$ is not consistent with $\mathbf{u}_1$ anymore (see Figure 8 (c)). This reasoning can be repeated $\bar{k}$ times to build the feasible control sequence. Note that the statement Fall$(\mathbf{u}_k, \mathcal{T}(\mathbf{u}_{k-1}, \mathbf{u}_k))$ at Step 6 amounts to performing the following statements:

$$\begin{array}{lcl} \mathbb{Z}_{k-1} & := & \mathbb{Z}_{k-1} \cap \pi_{\mathbf{u}}^{-1}(\mathbb{U}_{k-1}); \\ \mathbb{X}_k & := & \mathbb{X}_k \cap \mathbf{f}(\mathbb{Z}_{k-1}); \\ \mathbb{Z}_k & := & \mathbb{Z}_k \cap \pi_{\mathbf{x}}^{-1}(\mathbb{X}_k); \\ \mathbb{U}_k & := & \mathbb{U}_k \cap \pi_{\mathbf{u}}(\mathbb{Z}_k); \end{array} \tag{29}$$

Note that since $\mathcal{T}(\mathbf{u}_{k-1}, \mathbf{u}_k)$ is a chain, Fall$(\mathbf{u}_k, \mathcal{T}(\mathbf{u}_{k-1}, \mathbf{u}_k))$ could also be replaced by Climb$(\mathbf{u}_{k-1}, \mathcal{T}(\mathbf{u}_{k-1}, \mathbf{u}_k))$ at Step 6 of OpenLoopControl.

Remark 6 *The conceptual algorithm* OpenLoopControl*, where Step 6 is replaced by the sequence (29) is similar to that of Bertsekas [4] which has been obtained without taking into account that the graph associated with the constraints is a tree. However, a practical*

implementation of this algorithm requires some computations with sets (such as $\mathbf{f}(\mathbb{Z}_{k-1})$ *and* $\pi_{\mathbf{u}}(\mathbb{Z}_k)$*), which can be performed (in an approximate but guaranteed way) only with interval methods, except for particular systems.*

4.2 Closed-loop set controller

Assume now that the initial state vector is not known anymore and that at time k, a measurement $\widetilde{\mathbf{y}}_k$ of $\mathbf{y}_k$ is collected. In a bounded-error context, to each measurement is associated a set-membership relation of the form $\mathbf{y}_k \in \mathcal{I}(\widetilde{\mathbf{y}}_k)$ where $\mathcal{I}$ is the *interpretation function* which associates to the noisy data $\widetilde{\mathbf{y}}_k$ a set $\mathcal{I}(\widetilde{\mathbf{y}}_k)$ to which the noise-free output $\mathbf{y}_k$ should belong. Thus, when the measurement $\widetilde{\mathbf{y}}_k$ is collected, the domain $\mathbb{Y}_k$ of $\mathbf{y}_k$ has to be contracted into $\mathbb{Y}_k \cap \mathcal{I}(\widetilde{\mathbf{y}}_k)$.

Assume that at time k the state vector is known to belong to $\mathbb{X}_k$ ($\neq \emptyset$). The closed loop control problem at time k amounts to finding $\mathbf{u}_k \in \mathbb{U}_k$ such that all future specifications can be satisfied. The following algorithm takes into account that at time k a new measurement vector is collected, in order to compute a feasible control sequence.

Algorithm ClosedLoopControl(in: $\mathbb{X}_0, \mathbb{U}_0, \mathbb{Y}_1, \ldots$; out: $\widetilde{\mathbf{u}}_0, \ldots, \widetilde{\mathbf{u}}_{\bar{k}-1}$)

```
1   Z_0 := R^{n_z}; ...; Z_{k̄-1} := R^{n_z};
2   Fall(u_0, G);
3   for k := 0 to k̄ - 2
4         if U_k = ∅ return ("fail");
5         choose ũ_k ∈ U_k;  U_k := {ũ_k};
6         apply the control ũ_k to the system;
7         wait for a measurement ỹ_{k+1}; Y_{k+1} := I(ỹ_{k+1}) ∩ Y_{k+1};
8         Fall(u_{k+1}, T(u_k, u_{k+1}, y_{k+1}));
9   end for;
10  choose ũ_{k̄-1} ∈ U_{k̄-1}.
```

Comments: This algorithm is similar to OpenLoopControl. The main difference is that measurements are collected at time k and contract the domains in an unpredictable way. After Step 2, the consistency graph is given by Figure 9 (a). If the consistency domain associated with $\mathbf{u}_0$ is empty, then nothing can be done to find a feasible control sequence and a failure is returned at Step 4. At Step 5, a $\widetilde{\mathbf{u}}_0$ is chosen in $\mathbb{U}_0$. It can be chosen in an optimal way, but an optimal strategy often corresponds to choosing a point that is on the border of $\mathbb{U}_0$ and increases the probability of failure at the next steps. A

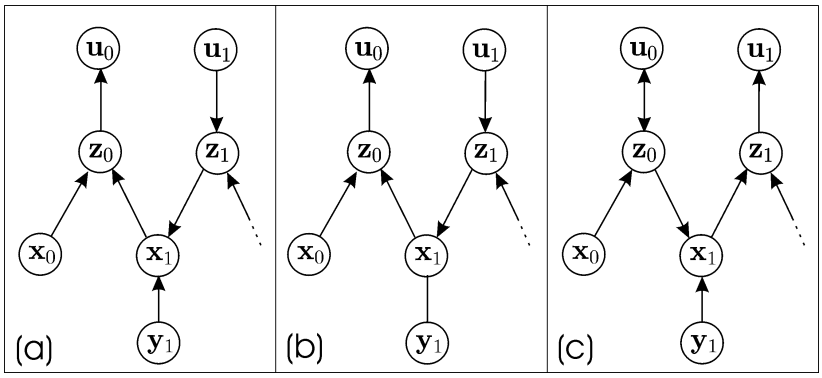

Figure 9: Principle of the algorithm ClosedLoopControl

robust strategy may correspond to choosing a point that is deep inside $\mathbb{U}_0$. Once $\mathbf{u_0}$ has been instantiated, the new domain for $\mathbb{U}_0$ becomes equal to the singleton $\{\widetilde{\mathbf{u}}_0\}$ at Step 5. At this level, the consistency graph is still given by Figure 9 (a). After applying to the system the control $\widetilde{\mathbf{u}}_0$, we wait for the measurement $\widetilde{\mathbf{y}}_1$. This translates into a contraction of $\mathbb{Y}_1$ and $\mathbf{x}_1$ is not anymore consistent with $\mathbf{y}_1$. The consistency graph is now given by Figure 9 (b). Step 8 computes the consistency domain for $\mathbf{u}_1$ and the consistency graph is now given by Figure 9 (c). Note that Step 8 translates into the following statements.

$$\begin{array}{lcl} \mathbb{Z}_k & := & \mathbb{Z}_k \cap \pi_{\mathbf{u}}^{-1}(\mathbb{U}_k)\,; \\ \mathbb{X}_{k+1} & := & \mathbb{X}_{k+1} \cap \mathbf{f}(\mathbb{Z}_k) \cap \mathbf{g}^{-1}(\mathbb{Y}_{k+1})\,; \\ \mathbb{Z}_{k+1} & := & \mathbb{Z}_{k+1} \cap \pi_{\mathbf{x}}^{-1}(\mathbb{X}_{k+1})\,; \\ \mathbb{U}_{k+1} & := & \mathbb{U}_{k+1} \cap \pi_{\mathbf{u}}(\mathbb{Z}_{k+1})\,; \end{array} \tag{30}$$

Step 3 to 9 are repeated several times, until the index k becomes equal to $\bar{k} - 2$.

5 Test cases

5.1 Test case 1

Consider the *timed-event graph* illustrated by the Petri net (see [23] for a survey) of Figure 10.

In this system, tokens (represented by black points) circulate following the directions of the arrows. Transverse segments correspond to *transitions* and circles correspond to *places*. At time k, a transition is *fired* when all its upstream places contain at least one

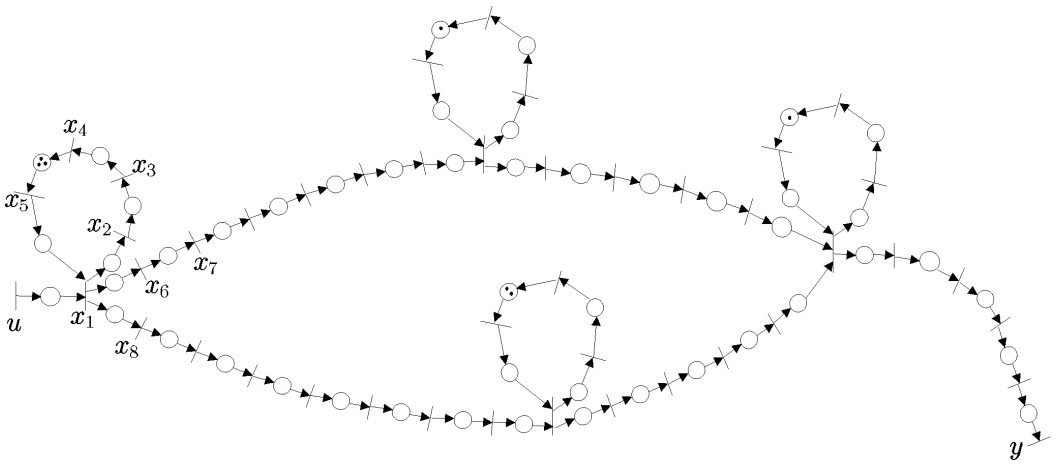

Figure 10: Discrete-time system to be controlled

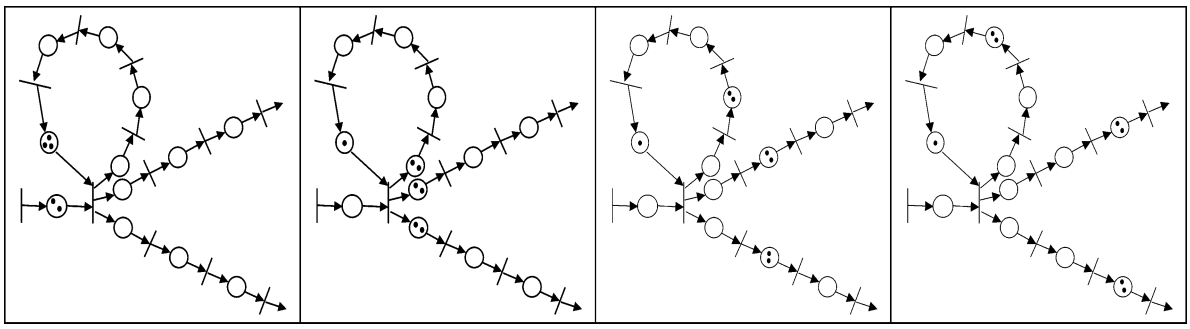

Figure 11: Evolution of the Petri net for $k = 1, 2, 3$ and 4

token. In such a case, one token is removed from each of the upstream places and one token is added to all downstream places. This operation takes place until at least one of the upstream places becomes empty. At time $k = 0$, there exists 7 tokens in the system (see Figure 10). Assume that at time $k = 1$ two tokens have been inserted in the Petri net through the control transition u. The evolution of the system for $k = 1, 2, 3$ and 4 is illustrated by Figure 11.

Denote by x_k^i (u_k and y_k, respectively) the number of tokens that have crossed the transition x_i (u and y, respectively) at time k. For instance, in the situation represented in Figure 11, we have $x_0^1 = 0, x_1^1 = 0$ and $x_k^1 = 2$, for $k \geq 2$. Note the sequences x_k^i, u_k and y_k increase with k. The evolution of the system can be described by the following state-space equations (see [6] for more details on the state-space representation of timed-event

graph):

$$\begin{pmatrix} x_k^1 \\ x_k^2 \\ x_k^3 \\ x_k^4 \\ x_k^5 \\ x_k^6 \\ x_k^7 \\ x_k^8 \\ \vdots \\ x_k^{43} \end{pmatrix} = \begin{pmatrix} \min(u_{k-1}, x_{k-1}^5) \\ x_{k-1}^1 \\ x_{k-1}^2 \\ x_{k-1}^3 \\ x_{k-1}^4 + 3 \\ x_{k-1}^1 \\ x_{k-1}^6 \\ x_{k-1}^1 \\ \vdots \\ x_{k-1}^{42} \end{pmatrix}. \tag{31}$$

We shall assume that at time $k = 0$ no transition has been fired, i.e., for all i, $x_0^i = 0$. The output to be considered is the number of tokens that leave the system. This leads us to the observation equation

$$y_k = x_k^{43}, \tag{32}$$

where x_k^{43} corresponds to the rightmost transition.

Let us now illustrate the methodology followed by OpenLoopControl to find a control sequence which satisfy some given specifications. For instance, assume that it is required that at time $k = 24$ two tokens and for $k = 33$ three tokens should have left the system. Taking into account the fact that y_k is an increasing sequence, these specifications yield the following domains for y_k.

k	$0 \rightarrow 23$	$24 \rightarrow 32$	$33 \rightarrow 34$
$\mathbb{Y}_k$	$[0, \infty[$	$[2, \infty[$	$[3, \infty[$

(33)

All other domains are taken as $]-\infty, \infty[$. Fall$(u_0, \mathcal{G})$ provides the following domains for

u_k and y_k.

k	$\widehat{\mathbb{U}}_k$	$\widehat{\mathbb{Y}}_k$
0	$[0, \infty[$	$[0, \infty[$
$1 \rightarrow 4$	$[1, \infty[$	$[0, \infty[$
$5 \rightarrow 13$	$[2, \infty[$	$[0, \infty[$
$14 \rightarrow 16$	$[3, \infty[$	$[0, \infty[$
$17 \rightarrow 19$	$]-\infty, \infty[$	$[0, \infty[$
$20 \rightarrow 23$	$]-\infty, \infty[$	$[1, \infty[$
$24 \rightarrow 32$	$]-\infty, \infty[$	$[3, \infty[$
$33 \rightarrow 34$	$]-\infty, \infty[$	$[3, \infty[$

(34)

Note that, since all initial domains are corners (except $\mathbf{x_0}$ which is known and can be removed from $\mathcal{V}$), and since all set operations involved in FALL$(u_0, \mathcal{G})$ are corner-conservative, FALL$(u_0, \mathcal{G})$ can be implemented in a very efficient way (less than 0.1 second in a Pentium III). Now, at Step 5, OPENLOOPCONTROL chooses any u_0 in $\mathbb{U}_0$. At Step 6, FALL$(u_1, \mathcal{T}(u_0, u_1))$ translates into the sequence of statements

$$\begin{array}{ll} 1 & \mathbb{Z}_0 = \pi_{\mathbf{x}}^{-1}(\mathbb{X}_0) \cap \pi_u^{-1}(\mathbb{U}_0) \cap \mathbb{Z}_0; \\ 2 & \mathbb{X}_1 = \mathbf{f}_1(\mathbb{Z}_0) \cap \mathbb{X}_1; \\ 3 & \mathbb{Z}_1 := \pi_{\mathbf{x}}^{-1}(\mathbb{X}_1) \cap \mathbb{Z}_1; \\ 4 & \mathbb{U}_1 := \pi_u(\mathbb{Z}_1). \end{array} \tag{35}$$

Now, before this sequence, $\mathbb{Z}_1$ is the corner: $\mathbb{Z}_1 = \mathbb{X}_1 \times \mathbb{U}_1$. After Step 3, $\mathbb{Z}_1 := \pi_{\mathbf{x}}^{-1}(\mathbb{X}_1') \times \mathbb{U}_1$ where $\mathbb{X}_1' \subset \mathbb{X}_1$. Its projection onto the u_1-space remains the same. Therefore, any instantiation of u_0 does not change the domain for u_1. This reasoning made for $k = 0$ can also be done for $k = 1, 2, \ldots$. This shows that the set of all feasible control sequences $(u_0, \ldots, u_{34})$ in a corner of $\mathbb{R}^{35}$ and that any arbitrary choice of u_k at Step 5 of OPENLOOPCONTROL will not restrict the choice for $u_{k+1}, u_{k+2}, \ldots$ This means that, for our testcase, the u_k's can be chosen independently in their domains immediately after Step 2. Note that the corner of all feasible control sequences contains feasible sequences that are not monotonic, but which satisfy all constraints. Of course, such decreasing sequences have no physical meaning and should not be chosen. A as-late-as-possible strategy would lead to the sequence

$$(u_0, \ldots, u_{34}) = (0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, \ldots, 3) \tag{36}$$

which is the smallest feasible sequence. The same sequence would have been obtained using the residuation theory applied to optimal control of timed-event graph [19, 6].

5.2 Test case 2

Consider the discrete-time system described by the following state-space equations

$$\begin{cases} x_k^1 &= \frac{1}{2} x_{k-1}^1, \\ x_k^2 &= x_{k-1}^2 u_{k-1} + 1, \\ y_k &= \mathrm{sqr}\left(x_k^1\right) + \mathrm{sqr}\left(x_k^2\right), \end{cases} \tag{37}$$

where $k \in \{1, \ldots, 5\}$. The required domains for the elements of the state vectors and the output are $\mathbb{X}_1 = \mathbb{X}_2 = \mathbb{X}_3 = \mathbb{X}_4 = \mathbb{X}_5 = [-1, 1] \times [-1, 1]$, and $\mathbb{Y}_1 = \mathbb{Y}_2 = \mathbb{Y}_3 = \mathbb{Y}_4 = \mathbb{Y}_5 = [-1, 1]$. The initial domains for the controls are $\mathbb{U}_0 = \mathbb{U}_1 = \mathbb{U}_2 = \mathbb{U}_3 = \mathbb{U}_4 = [0, 5]$. The initial state is taken as $\mathbf{x_0} = (1, -1)$.

For this test-case, the computation of $f(\mathbb{X})$, $f^{-1}(\mathbb{Y})$ or $\mathbb{X} \cap \mathbb{Y}$, needed by FALL and CLIMB, can only be approximated. This can be done, for instance, by using the *AQCS solver* (*approximate quantified constraint solving)* developed by one of the authors [25, 26]. AQCS computes inner and outer approximations of sets with an arbitrary accuracy and implements set computation using such approximations.

Using AQCS, OPENLOOPCONTROL computes the following feasible control sequence in about one second on a 500MhZ Linux machine. The feasible control sequence and the corresponding states and outputs are given by the following table (rounded to 3 digits).

k	0	1	2	3	4	5
u_k	1.866025	2.272732	2.05	1.96	0.5	
x_k^1	1,	0.5	0.25	0.125	0.0625	0.03125
x_k^2	-1	-0.866	-0.968	-0.985	-0.93	0.535
y_k	2	1	0.9995	0.986	0.869	0.2872

6 Robust control

6.1 Robust control problem

In a robust control and a bounded-error contexts, it is generally assumed that bounded perturbations $\mathbf{w}_k$ occur in the system and that the initial state vector $\mathbf{x_0}$ is unknown. The robust control problem amounts in finding a feasible sequence $\mathbf{u_0}, \ldots, \mathbf{u}_{\bar{k}}$ which satisfies all required specifications for all feasible perturbations $\mathbf{w}_k$ and for all feasible initial states

$\mathbf{x_0}$. This problem is much more difficult than the non-robust control problem proposed in Section 1, since the quantifier $\forall$ now occurs in the formulation of the constraints.

The equations (1) should now be written

$$\mathbf{x}_k \;=\; \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) \qquad k \in \{1, \ldots, \bar{k}\} \tag{38}$$

where $\mathbf{w}_k \in \mathbb{R}^{n_\mathrm{w}}$ is the perturbation vector which assumed to belong to domain $\mathbb{W}_k$. The initial state vector $\mathbf{x_0}$ is assumed to belong to the set $\mathbb{X}_\mathbf{0}$. For simplicity, we assume that no output specification are required. The output equation has thus been removed. Recall that the control and state vectors $\mathbf{u}_k$, $\mathbf{x}_k$ are all assumed to belong to known domains $\mathbb{U}_k$ and $\mathbb{X}_k$ which may be arbitrarily large.

The robust control problem can be formulated as follows.

Robust control problem : find $\mathbf{u_0} \in \mathbb{U}_0, \mathbf{u}_1 \in \mathbb{U}_1, \ldots, \mathbf{u}_{\bar{k}-1} \in \mathbb{U}_{\bar{k}-1}$ such that

$$\left\{ \begin{array}{l} \forall \mathbf{w_0} \in \mathbb{W}_0, \ldots, \forall \mathbf{w}_{\bar{k}-1} \in \mathbb{W}_{\bar{k}-1}, \\ \forall \mathbf{x_0} \in \mathbb{X}_0, \exists \mathbf{x}_1 \in \mathbb{X}_1, \ldots, \exists \mathbf{x}_{\bar{k}} \in \mathbb{X}_{\bar{k}}, \\ \mathbf{x}_1 = \mathbf{f}(\mathbf{x_0}, \mathbf{u_0}, \mathbf{w_0}), \ldots, \mathbf{x}_{\bar{k}} = \mathbf{f}(\mathbf{x}_{\bar{k}-1}, \mathbf{u}_{\bar{k}-1}, \mathbf{w}_{\bar{k}-1}) \end{array} \right. \tag{39}$$

We shall assume here that we are able to compute with sets (*i.e.*, to compute the reciprocal or the direct image of a set by a function, to project a set, ...). Let us recall that sometimes, (e.g., when sets are polygons and functions are linear), set computation can be performed exactly. When the sets involved have low dimensions it is possible (for most practical systems) to approximate the set operations by using interval solvers. In this section, we shall propose a set algorithm that will be able to solve our robust control problem. We shall take into account the structure of the problem in order to avoid the manipulation of high dimensional sets, *i.e.*, the dimension of the sets should not depend on $\bar{k}$.

6.2 Notions of first-order logic

This small section presents some notations and recalls basic notions related to the first-order logic (see [11] for more information on the topic). These notions will be needed to transform our robust control problem into a sequence of set operations, where the dimensions of all sets are $\bar{k}$-independent.

Roughly speaking, a first-order logic involves *predicates* (such as $x_1 \geq x_2 + 1$) and *quantifiers* $\exists$ and $\forall$. The *free variables* of a first-order formula are those that are not associated to a quantifier.

For instance, Formula (39) is a first-order formula. Its $\bar{k}$ free variables are $\mathbf{u_0}, \mathbf{u}_1, \ldots, \mathbf{u}_{\bar{k}-1}$, its $2\bar{k}+1$ quantified variables are $\mathbf{w_0}, \ldots, \mathbf{w}_{\bar{k}-1}, \mathbf{x_0}, \mathbf{x}_1, \ldots, \mathbf{x}_{\bar{k}}$ and its $\bar{k}$ predicates are "$\mathbf{x}_1 = \mathbf{f}(\mathbf{x_0}, \mathbf{u_0}, \mathbf{w_0}), \ldots, \mathbf{x}_{\bar{k}} = \mathbf{f}(\mathbf{x}_{\bar{k}-1}, \mathbf{u}_{\bar{k}-1}, \mathbf{w}_{\bar{k}-1})$".

If, in a first-order formula, all quantifiers appear in the front, then the formula is said to be in a *prenex form*. This is the case for Formula (39).

If $\alpha(a)$, $\beta(a,b)$ are two first-order formulas with free variables a, $\{a,b\}$, respectively, and if f is a function, we have the following *right shift rules :*

$$\begin{array}{lcl} \text{(s1)} \quad \exists a \in \mathbb{A}, \forall b \in \mathbb{B}, \alpha(a), \beta(a,b) & \Leftrightarrow & \exists a \in \mathbb{A}, \alpha(a), \forall b \in \mathbb{B}, \beta(a,b) \\ \text{(s2)} \quad \forall a \in \mathbb{A}, \exists b \in \mathbb{B}, \alpha(a), \beta(a,b) & \Leftrightarrow & \forall a \in \mathbb{A}, \alpha(a), \exists b \in \mathbb{B}, \beta(a,b) \end{array} \tag{40}$$

The purpose of these rules is to move the quantifiers on the right of the formula. In order to show how we shall use these rules, let us prove the following equivalence

$$\exists a \in \mathbb{A}, \forall b \in \mathbb{B}, \alpha(b,c), \beta(a,c) \Leftrightarrow \forall b \in \mathbb{B}, \alpha(b,c), \exists a \in \mathbb{A}, \beta(a,c). \tag{41}$$

The left formula $\exists a \in \mathbb{A}, (\forall b \in \mathbb{B}, \alpha(b,c)), \beta(a,c)$ can be rewritten as $\exists a \in \mathbb{A}, \gamma(c), \beta(a,c)$. From (s1), we get $\gamma(c), \exists a \in \mathbb{A}, \beta(a,c)$ or equivalently $\forall b \in \mathbb{B}, \alpha(b,c), \exists a \in \mathbb{A}, \beta(a,c)$.

Define the following operators for two sets $\mathbb{A}$ and $\mathbb{B}$:

$$\mathbb{A} \backslash \mathbb{B} \triangleq \{a \in \mathbb{A}, a \notin \mathbb{B}\}. \tag{42}$$

The following rules make it possible to rewrite a formula with quantifiers into a predicate.

$$\begin{array}{lcl} \text{(e1)} \quad \exists a \in \mathbb{A}, a = f(b) & \Leftrightarrow & b \in f^{-1}(\mathbb{A}) \\ \text{(e2)} \quad a \in \mathbb{A}, \exists b \in \mathbb{B}, (a,b) \in \mathbb{C} & \Leftrightarrow & a \in \pi_a((\mathbb{A} \times \mathbb{B}) \cap \mathbb{C}) \\ \text{(e3)} \quad a \in \mathbb{A}, \forall b \in \mathbb{B}, (a,b) \in \mathbb{C} & \Leftrightarrow & a \in \mathbb{A} \backslash \pi_a((\mathbb{A} \times \mathbb{B}) \backslash \mathbb{C}) \\ \text{(e4)} \quad \forall a \in \mathbb{A}, \exists b \in \mathbb{B}, b = f(a), \alpha(b,c) & \Leftrightarrow & \forall b \in \mathbb{B} \cap f(\mathbb{A}), \alpha(b,c) \end{array} \tag{43}$$

where π denotes the projection operator with respect to the variable a. Figure 12 give a graphical interpretation of equivalences (e2) and (e3).

6.3 Algorithm

Theorem 1 *The robust control problem can be solved by the following set algorithm.*

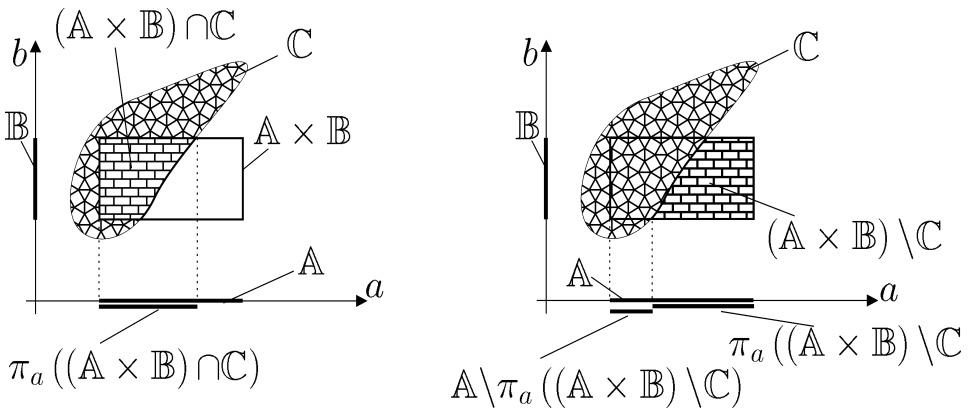

Figure 12: Illustration of the equivalences (e2) and (e3)

Algorithm RobustControl(in: $\mathbb{X}_{\mathbf{0}}, \mathbb{U}_{\mathbf{0}}, \mathbb{Y}_1, \ldots, \mathbb{X}_{\bar{k}}, \mathbb{U}_{\bar{k}-1}, \mathbb{Y}_{\bar{k}}$; out $\widetilde{\mathbf{u}}_{\mathbf{0}}, \ldots, \widetilde{\mathbf{u}}_{\bar{k}-1}$)

1 $\quad\overleftarrow{\mathbb{X}}_{\bar{k}} := \mathbb{X}_{\bar{k}}$;
2 $\quad$for $k := \bar{k}$ downto 2
3 $\quad\quad\overleftarrow{\mathbb{X}}_{k-1} := \mathbb{X}_k \cap \pi_{\mathbf{x}}\left(\mathbb{X}_{k-1}\times\mathbb{U}_{k-1} \setminus \pi_{\mathbf{x},\mathbf{u}}\left(\mathbb{X}_{k-1}\times\mathbb{U}_{k-1}\times\mathbb{W}_{k-1} \setminus \mathbf{f}^{-1}(\overleftarrow{\mathbb{X}}_k)\right)\right)$;
4 $\quad$endfor
5 $\quad\widehat{\mathbb{U}}_{\mathbf{0}} := \mathbb{U}_{\mathbf{0}} \setminus \pi_{\mathbf{u}}\left(\mathbb{X}_{\mathbf{0}}\times\mathbb{U}_{\mathbf{0}} \cap \pi_{\mathbf{x},\mathbf{u}}\left(\mathbb{X}_{\mathbf{0}}\times\mathbb{U}_{\mathbf{0}}\times\mathbb{W}_{\mathbf{0}} \setminus \mathbf{f}^{-1}(\overleftarrow{\mathbb{X}}_1)\right)\right)$;
6 $\quad$if $\widehat{\mathbb{U}}_{\mathbf{0}} = \emptyset$ return "The robust control problem has no solution"; end;
7 $\quad$choose $\widetilde{\mathbf{u}}_{\mathbf{0}} \in \widehat{\mathbb{U}}_{\mathbf{0}}$;
8 $\quad\overrightarrow{\mathbb{X}}_{\mathbf{0}} := \mathbb{X}_{\mathbf{0}}$;
9 $\quad$for $k := 1$ to $\bar{k}-1$
10 $\quad\quad\overrightarrow{\mathbb{X}}_k := \mathbb{X}_k \cap \mathbf{f}(\overrightarrow{\mathbb{X}}_{k-1}, \widetilde{\mathbf{u}}_{k-1}, \mathbb{W}_{k-1})$;
11 $\quad\quad\widehat{\mathbb{U}}_k := \mathbb{U}_k \setminus \pi_{\mathbf{u}}\left(\overrightarrow{\mathbb{X}}_k\times\mathbb{U}_k\times\mathbb{W}_k \setminus \mathbf{f}^{-1}(\overleftarrow{\mathbb{X}}_{k+1})\right)$;
12 $\quad\quad$choose $\widetilde{\mathbf{u}}_k \in \widehat{\mathbb{U}}_k$;
13 $\quad$endfor

The set $\overleftarrow{\mathbb{X}}_k$ represents the set of all state vectors at time k such that if $\mathbf{x}_k \in \overleftarrow{\mathbb{X}}_k$ then for all feasible future perturbations, it is possible to find a control such that all future specifications are satisfied.

The set $\overrightarrow{\mathbb{X}}_k$ represents all possible (i.e., for all feasible initial state and for all feasible past perturbations) state vectors that can be reached at time k if the past controls $\mathbf{u}_{\mathbf{0}} = \widetilde{\mathbf{u}}_{\mathbf{0}}, \ldots, \mathbf{u}_{k-1} = \widetilde{\mathbf{u}}_{k-1}$ have been chosen.

If we assume that the set operations can be executed, then, this algorithm will find (if exists) the robust control sequence. Recall that, in general (but not always), the set operations can only be approximated.

The complexity of most set operations is exponential with respect to the dimensions of the sets. Now, in the algorithm RobustControl all sets involved have a dimension smaller than $n_{\mathbf{x}} + n_{\mathbf{w}} + n_{\mathbf{u}}$. Therefore, the complexity of the algorithm can be considered as linear with respect to $\bar{k}$ and exponential with respect to $n_{\mathbf{x}} + n_{\mathbf{w}} + n_{\mathbf{u}}$.

6.4 Proof of the theorem

The proof is by induction. The main idea is to transform the quantified formula into set computation. We shall first explain how $\mathbf{u_0}$ can be found. Then, we shall see how $\mathbf{u}_k$ can be found from the knowledge of $\mathbf{u_0}, \ldots, \mathbf{u}_{k-1}$.

Robust control problem at time $k = 0$: Find $\mathbf{u_0} \in \mathbb{U}_0$ such that

$$F(\mathbf{u_0}) : \begin{cases} \exists \mathbf{u}_1 \in \mathbb{U}_1, \ldots, \exists \mathbf{u}_{\bar{k}-1} \in \mathbb{U}_{\bar{k}-1}, \\ \forall \mathbf{w_0} \in \mathbb{W}_0, \ldots, \forall \mathbf{w}_{\bar{k}-1} \in \mathbb{W}_{\bar{k}-1}, \\ \forall \mathbf{x_0} \in \mathbb{X}_0, \exists \mathbf{x}_1 \in \mathbb{X}_1, \ldots, \exists \mathbf{x}_{\bar{k}} \in \mathbb{X}_{\bar{k}}, \\ \mathbf{x}_1 = \mathbf{f}(\mathbf{x_0}, \mathbf{u_0}, \mathbf{w_0}), \ldots, \mathbf{x}_{\bar{k}} = \mathbf{f}(\mathbf{x}_{\bar{k}-1}, \mathbf{u}_{\bar{k}-1}, \mathbf{w}_{\bar{k}-1}). \end{cases} \tag{44}$$

Note that the only free variable of the first-order formula $F(\mathbf{u_0})$ is $\mathbf{u_0}$. Successive applications of the right shift rules (40) transform $F(\mathbf{u_0})$ into

$$\begin{cases} 1 & \forall \mathbf{x_0} \in \mathbb{X}_0, & \forall \mathbf{w_0} \in \mathbb{W}_0, & \exists \mathbf{x}_1 \in \mathbb{X}_1, & \mathbf{x}_1 = \mathbf{f}(\mathbf{x_0}, \mathbf{u_0}, \mathbf{w_0}) \\ 2 & \exists \mathbf{u}_1 \in \mathbb{U}_1, & \forall \mathbf{w}_1 \in \mathbb{W}_1, & \exists \mathbf{x}_2 \in \mathbb{X}_2, & \mathbf{x}_2 = \mathbf{f}(\mathbf{x}_1, \mathbf{u}_1, \mathbf{w}_1) \\ 3 & \ldots & \ldots & \ldots & \ldots \\ 4 & \exists \mathbf{u}_{\bar{k}-1} \in \mathbb{U}_{\bar{k}-1}, & \forall \mathbf{w}_{k-1} \in \mathbb{W}_{\bar{k}-1}, & \exists \mathbf{x}_{\bar{k}} \in \mathbb{X}_{\bar{k}}, & \mathbf{x}_{\bar{k}} = \mathbf{f}(\mathbf{x}_{\bar{k}-1}, \mathbf{u}_{\bar{k}-1}, \mathbf{w}_{\bar{k}-1}). \end{cases} \tag{45}$$

Let us eliminate all quantifiers in a backward way (i.e., the last first). From (e1) of (43), we get that

$$\exists \mathbf{x}_{\bar{k}} \in \mathbb{X}_{\bar{k}}, \mathbf{x}_{\bar{k}} = \mathbf{f}(\mathbf{x}_{\bar{k}-1}, \mathbf{u}_{\bar{k}-1}, \mathbf{w}_{\bar{k}-1}) \Leftrightarrow (\mathbf{x}_{\bar{k}-1}, \mathbf{u}_{\bar{k}-1}, \mathbf{w}_{\bar{k}-1}) \in \mathbf{f}^{-1}(\mathbb{X}_{\bar{k}}) \tag{46}$$

which eliminates the last quantifier. The previous quantifier $(\forall \mathbf{w}_{k-1})$ is now eliminated as follows :

$$\begin{array}{ll} & \forall \mathbf{w}_{\bar{k}-1} \in \mathbb{W}_{\bar{k}-1}, (\mathbf{x}_{\bar{k}-1}, \mathbf{u}_{\bar{k}-1}, \mathbf{w}_{\bar{k}-1}) \in \mathbf{f}^{-1}(\mathbb{X}_{\bar{k}}) \\ \overset{\text{(e3)}}{\Leftrightarrow} & (\mathbf{u}_{\bar{k}-1}, \mathbf{x}_{\bar{k}-1}) \in (\mathbb{U}_{\bar{k}-1} \times \mathbb{X}_{\bar{k}-1}) \setminus \pi_{\mathbf{x},\mathbf{u}}(\mathbb{X}_{\bar{k}-1} \times \mathbb{U}_{\bar{k}-1} \times \mathbb{W}_{\bar{k}-1} \setminus \mathbf{f}^{-1}(\mathbb{X}_{\bar{k}})) \end{array} \tag{47}$$

The elimination of the $(\exists \mathbf{u}_{\bar{k}-1})$ quantifier is performed using rule (e2). We get that the line 4 of (45) (the only free variable of which is $\mathbf{x}_{\bar{k}-1}$) is equivalent to

$$\mathbf{x}_{\bar{k}-1} \in \overleftarrow{\mathbb{X}}_{\bar{k}-1}, \tag{48}$$

where

$$\overleftarrow{\mathbb{X}}_{\bar{k}-1} \triangleq \mathbb{X}_{\bar{k}} \cap \pi_{\mathbf{x}} \left((\mathbb{U}_{\bar{k}-1} \times \mathbb{X}_{\bar{k}-1}) \setminus \pi_{\mathbf{x},\mathbf{u}} \left((\mathbb{X}_{\bar{k}-1} \times \mathbb{U}_{\bar{k}-1} \times \mathbb{W}_{\bar{k}-1}) \setminus \mathbf{f}^{-1}(\mathbb{X}_{\bar{k}}) \right) \right). \tag{49}$$

The procedure is performed until all quantifiers have been removed. We get that $F(\mathbf{u_0})$ is equivalent to the predicate '$\mathbf{u_0} \in \widehat{\mathbb{U}}_0$' where $\widehat{\mathbb{U}}_0$ is the set computed Steps 1 to 5 of the algorithm RobustControl.

Robust control problem at time k:

Assume that $\mathbf{u}_0, \ldots, \mathbf{u}_{k-1}$ have been chosen to be equal to $\widetilde{\mathbf{u}}_0, \ldots, \widetilde{\mathbf{u}}_{k-1}$. The problem at time k is that of finding $\mathbf{u}_k \in \mathbb{U}_k$ such that

$$F(\mathbf{u}_k) : \left\{ \begin{array}{l} \exists \mathbf{u}_{k+1} \in \mathbb{U}_{k+1}, \ldots, \exists \mathbf{u}_{\bar{k}-1} \in \mathbb{U}_{\bar{k}-1}, \\ \forall \mathbf{w}_0 \in \mathbb{W}_0, \ldots, \forall \mathbf{w}_{\bar{k}-1} \in \mathbb{W}_{\bar{k}-1}, \\ \forall \mathbf{x}_0 \in \mathbb{X}_0, \exists \mathbf{x}_1 \in \mathbb{X}_1, \ldots, \exists \mathbf{x}_{\bar{k}} \in \mathbb{X}_{\bar{k}}, \\ \mathbf{x}_1 = \mathbf{f}(\mathbf{x}_0, \widetilde{\mathbf{u}}_0, \mathbf{w}_0), \ldots, \mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \widetilde{\mathbf{u}}_{k-1}, \mathbf{w}_{k-1}), \\ \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k), \ldots, \mathbf{x}_{\bar{k}} = \mathbf{f}(\mathbf{x}_{\bar{k}-1}, \mathbf{u}_{\bar{k}-1}, \mathbf{w}_{\bar{k}-1}). \end{array} \right. \tag{50}$$

Note that in the first-order formula $F(\mathbf{u}_k)$, the only unquantified variable is $\mathbf{u}_k$ (since $\widetilde{\mathbf{u}}_0, \ldots, \widetilde{\mathbf{u}}_{k-1}$ have already been instantiated, they should not be considered as variable anymore). Iterative application of the right shift rules yields that $F(\mathbf{u}_k)$ is equivalent to

$$\left\{ \begin{array}{llllll} 1 & \forall \mathbf{x}_0 \in \mathbb{X}_0, & \forall \mathbf{w}_0 \in \mathbb{W}_0, & \exists \mathbf{x}_1 \in \mathbb{X}_1, & \mathbf{x}_1 = \mathbf{f}(\mathbf{x}_0, \widetilde{\mathbf{u}}_0, \mathbf{w}_0), \\ 2 & & \forall \mathbf{w}_1 \in \mathbb{W}_1, & \exists \mathbf{x}_2 \in \mathbb{X}_2, & \mathbf{x}_2 = \mathbf{f}(\mathbf{x}_1, \widetilde{\mathbf{u}}_1, \mathbf{w}_1), \\ 3 & \ldots & \ldots & \ldots & \ldots \\ 4 & & \forall \mathbf{w}_{k-1} \in \mathbb{W}_{k-1}, & \exists \mathbf{x}_k \in \mathbb{X}_k, & \mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \widetilde{\mathbf{u}}_{k-1}, \mathbf{w}_{k-1}), \\ 5 & & \forall \mathbf{w}_k \in \mathbb{W}_k, & \exists \mathbf{x}_{k+1} \in \mathbb{X}_{k+1}, & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k), \\ 6 & \exists \mathbf{u}_{k+1} \in \mathbb{U}_{k+1}, & \forall \mathbf{w}_{k+1} \in \mathbb{W}_{k+1}, & \exists \mathbf{x}_{k+2} \in \mathbb{X}_{k+2}, & \mathbf{x}_{k+2} = \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}, \mathbf{w}_{k+1}), \\ 7 & \ldots & \ldots & \ldots & \ldots \\ 8 & \exists \mathbf{u}_{\bar{k}-1} \in \mathbb{U}_{\bar{k}-1}, & \forall \mathbf{w}_{\bar{k}-1} \in \mathbb{W}_{\bar{k}-1}, & \exists \mathbf{x}_{\bar{k}} \in \mathbb{X}_{\bar{k}}, & \mathbf{x}_{\bar{k}} = \mathbf{f}(\mathbf{x}_{\bar{k}-1}, \mathbf{u}_{\bar{k}-1}, \mathbf{w}_{\bar{k}-1}). \end{array} \right. \tag{51}$$

We have shown that the subformula composed by lines 6,7 and 8 of Formula (51) is equivalent to '$\mathbf{x}_{k+1} \in \overleftarrow{\mathbb{X}}_{k+1}$'. Let us now eliminate the first quantifiers of Formula (51) in

a forward manner. From (e4), we get that

$$\begin{array}{rl} & \forall \mathbf{x_0} \in \mathbb{X}_0, \forall \mathbf{w_0} \in \mathbb{W}_0, \exists \mathbf{x}_1 \in \mathbb{X}_1, \mathbf{x}_1 = \mathbf{f}(\mathbf{x}_0, \widetilde{\mathbf{u}}_0, \mathbf{w}_0), \alpha\left(\mathbf{x}_1\right) \\ \Leftrightarrow & \forall \mathbf{x}_1 \in \mathbb{X}_1 \cap \mathbf{f}(\mathbb{X}_0, \widetilde{\mathbf{u}}_0, \mathbb{W}_0), \alpha\left(\mathbf{x}_1\right) \end{array} \tag{52}$$

where $\alpha\left(\mathbf{x}_1\right)$ is the subformula given by lines 2,3,...,8 of Formula (51). Therefore, Formula (51) can be replaced by

$$\left\{ \begin{array}{l} \forall \mathbf{x}_1 \in \mathbb{X}_1 \cap \mathbf{f}(\mathbb{X}_0, \widetilde{\mathbf{u}}_0, \mathbb{W}_0) \\ \forall \mathbf{w}_1 \in \mathbb{W}_1, \exists \mathbf{x}_2 \in \mathbb{X}_2, \mathbf{x}_2 = \mathbf{f}(\mathbf{x}_1, \widetilde{\mathbf{u}}_1, \mathbf{w}_1) \\ \ldots \\ \forall \mathbf{w}_{k-1} \in \mathbb{W}_{k-1}, \exists \mathbf{x}_k \in \mathbb{X}_k, \mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \widetilde{\mathbf{u}}_{k-1}, \mathbf{w}_{k-1}) \\ \forall \mathbf{w}_k \in \mathbb{W}_k, \exists \mathbf{x}_{k+1} \in \mathbb{X}_{k+1}, \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \\ \mathbf{x}_{k+1} \in \overleftarrow{\mathbb{X}}_{k+1}. \end{array} \right. \tag{53}$$

Applying (e4) of (43), $k-1$ times yields that Formula (53) is equivalent to

$$\left\{ \begin{array}{l} \forall \mathbf{x}_k \in \overrightarrow{\mathbb{X}}_k \\ \forall \mathbf{w}_k \in \mathbb{W}_k, \exists \mathbf{x}_{k+1} \in \mathbb{X}_{k+1}, \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k), \\ \mathbf{x}_{k+1} \in \overleftarrow{\mathbb{X}}_{k+1}. \end{array} \right. \tag{54}$$

where $\overrightarrow{\mathbb{X}}_k$ is defined as follows.

$$\begin{array}{rcl} \overrightarrow{\mathbb{X}}_0 & \triangleq & \mathbb{X}_0, \\ \overrightarrow{\mathbb{X}}_k & \triangleq & \mathbb{X}_k \cap \mathbf{f}(\overrightarrow{\mathbb{X}}_{k-1}, \widetilde{\mathbf{u}}_{k-1}, \mathbb{W}_{k-1}). \end{array} \tag{55}$$

From (e1), we get that Formula (54) is equivalent to

$$\forall \mathbf{x}_k \in \overrightarrow{\mathbb{X}}_k, \forall \mathbf{w}_k \in \mathbb{W}_k, \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \in \overleftarrow{\mathbb{X}}_{k+1}, \tag{56}$$

i.e.,

$$\forall \mathbf{x}_k \in \overrightarrow{\mathbb{X}}_k, \forall \mathbf{w}_k \in \mathbb{W}_k, (\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \in \mathbf{f}^{-1}(\overleftarrow{\mathbb{X}}_{k+1}). \tag{57}$$

Applying (e4), we get that the formula (57) (and thus Formula (50)) is equivalent to

$$\mathbf{u}_k \in \mathbb{U}_k \setminus \pi_{\mathbf{u}} \left((\overrightarrow{\mathbb{X}}_k \times \mathbb{U}_k \times \mathbb{W}_k) \setminus \mathbf{f}^{-1}(\overleftarrow{\mathbb{X}}_{k+1}) \right). \tag{58}$$

7 Conclusion

This paper proposes a new and general algorithm to compute a control sequence of a discrete-time system described by nonlinear state-space equations. The approach is based

on constraint propagation and takes advantage a the tree structure provided by the state formulation. The methodology has been illustrated by two test-cases. An extension to the case where uncertainties are involved in the system has also been considered. The special structure of state space equations have been taken into account in order to make the complexity of the algorithms linear with respect to the number $\bar{k}$ of samples.

The basic operations needed by our algorithms are difficult set operations such as computing the intersection of two sets, computing the direct image or the reciprocal image of a set. Computing exactly these set operations is sometimes possible (see Test-case 1), but in most situations, they can only be approximated. Interval algorithms (such as those used by the solver AQCS) can perform this task, but the computing time of existing algorithms is too large to allow the resolution of real-life problems.

However, our approach is general and made it possible to solve elementary problems in a reliable way for the first time. For instance, to our knowledge, Test-case 2 could not to be solved rigorously by other existing methods. Test-case 1 can be solved by using the residuation theory, but residuation only applies to *timed event graph systems*, which represent a tiny class of discrete-time systems.

Set algorithms seem to be appropriate to deal with the general (robust and non-robust) control problem. The main limitation of these algorithms is that, at the moment and for practical problems, their elementary set operations cannot be performed (with the required precision) in a reasonable computing time.

References

[1] B. R. Barmish. *New Tools for Robustness of Linear Systems.* MacMillan, New York, NY, 1994.

[2] F. Benhamou, F. Goualard, L. Granvilliers, and J. F. Puget. Revising hull and box consistency. In *Proceedings of the International Conference on Logic Programming*, pages 230–244, Las Cruces, NM, 1999.

[3] D. P. Bertsekas and I. B. Rhodes. On the minimax reachability of target sets and target tubes. *Automatica*, 7(2):233–241, 1971.

[4] D.P. Bertsekas. Infinite-time reachability of state-space regions by using feedback control. *IEEE Trans. Automatic Control*, 17(5):604–613, 1972.

[5] J. G. Cleary. Logical arithmetic. *Future Computing Systems*, 2(2):125–149, 1987.

[6] G. Cohen, P. Moller, J.P. Quadrat, and M. Viot. Algebraic Tools for the Performance Evaluation of Discrete Event Systems. *IEEE Proceedings: Special issue on Discrete Event Systems*, 77(1):39–58, January 1989.

[7] R. Dechter and J. Pearl. Tree-clustering for constraint networks. *Artificial Intelligence*, 38(3):353–366, April 1989.

[8] P. Dorato, R. Tempo, and G. Muscato. Bibliography on robust control. *Automatica*, 29(1):201–213, 1993.

[9] D.Q.Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: stability and optimality; survey paper. *Automatica*, 36:789–814, 2000.

[10] C. Durieu, B. Polyak, and E. Walter. Ellipsoidal state outer-bounding for MIMO systems via analytical techniques. In *Proceedings of the IMACS—IEEE–SMC CESA'96 Symposium on Modelling and Simulation*, volume 2, pages 843–848, Lille, France, 1996.

[11] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic.* Springer Verlag, 1984.

[12] M. Gondran and M. Minoux. *Graphes et algorithmes.* Eyrolles, Paris, 3e édition, 1995.

[13] L. Jaulin, M. Kieffer, I. Braems, and E. Walter. Guaranteed nonlinear estimation using constraint propagation on sets. *International Journal of Control*, 74(18):1772–1782, 2001.

[14] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer-Verlag, London, 2001.

[15] L. Jaulin and E. Walter. Global numerical approach to nonlinear discrete-time control. *IEEE Transactions on Automatic Control*, 42(6):872–875, 1997.

[16] A. Kurzhanski and I. Valyi. *Ellipsoidal Calculus for Estimation and Control*. Birkhäuser, Boston, MA, 1997.

[17] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[18] S. A. Malan, M. Milanese, and M. Taragna. Robust tuning for PID controllers with multiple performance specifications. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pages 2684–2689, Lake Buena Vista, 1994.

[19] E. Menguy, J.L. Boimond, and L. Hardouin. Just in time control of linear system in dioid: Cases of an update of reference input and uncontrollable input. *IEEE Transactions on Automatic Control*, 45(11):2155–2159, 2000.

[20] M. Milanese, J. Norton, H. Piet-Lahanier, and E. Walter, editors. *Bounding Approaches to System Identification*. Plenum Press, New York, NY, 1996.

[21] U. Montanari and F. Rossi. Constraint relaxation may be perfect. *Artificial Intelligence*, 48(2):143–170, 1991.

[22] R. E. Moore. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, PA, 1979.

[23] T. Murata. Petri nets : properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[24] J. P. Norton. Roles for deterministic bounding in environmental modeling. *Ecological Modelling*, 86:157–161, 1996.

[25] S. Ratschan. Approximate quantified constraint solving (AQCS). Available at: http://www.risc.uni-linz.ac.at/research/software/AQCS, 2000.

[26] Stefan Ratschan. Approximate quantified constraint solving by cylindrical box decomposition. *Reliable Computing*, 8(1):21–42, 2002.

[27] E. Walter and H. Piet-Lahanier. Exact recursive polyhedral description of the feasible parameter set for bounded-error models. *IEEE Transactions on Automatic Control*, 34(8):911–915, 1989.

[28] E. Walter and L. Pronzato. *Identification of Parametric Models from Experimental Data*. Springer-Verlag, London, UK, 1997.