Computing the Shape of the Image of a Multi-Linear Mapping Is Possible But Computationally Intractable: Theorems

Raúl Trejo and Vladik Kreinovich

Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
emails {rtrejo,vladik}@cs.utep.edu

Abstract

In systems without inertia (or with negligible inertia), a change in the values of control variables $x^{(1)}, \ldots, x^{(n)}$ leads to the immediate change in the state z of the system. In more precise terms, for such systems, every component z_i of the state vector $z = (z_1, \ldots, z_d)$ is a function of the control variables. When we know what state z we want to achieve, the natural question is: can we achieve this state, i.e., are there values of the control variables which lead to this very state?

The simplest possible functional dependence is described by *linear functions*. For such functions, the question of whether we can achieve a given state z reduces to the solvability of the corresponding system of linear equations; this solvability can be checked by using known (and feasible) algorithms from linear algebra.

Next in complexity is the case when instead of a linear dependence, we have a *multi-linear* dependence. In this paper, we show that for multi-linear functions, the controllability problem is, in principle, algorithmically solvable, but it is computationally hard (NP-hard).

1 Introduction

In systems without inertia (or with negligible inertia), a change in the values of control variables $x^{(1)} \in R^{d_1}, \ldots, x^{(n)} \in R^{d_n}$ leads to the immediate change in the state $z \in R^d$ of the system. In more precise terms, for such systems, every component z_i of the state vector $z = (z_1, \ldots, z_d)$ is a function of the variables $x^{(1)} = (x_1^{(1)}, \ldots, x_{d_1}^{(1)}), \ldots, x^{(n)} = (x_1^{(n)}, \ldots, x_{d_n}^{(n)})$, i.e.,

$$z_i = F_i(x^{(1)}, \ldots, x^{(n)}) = f_i(x_1^{(1)}, \ldots, x_{d_1}^{(1)}, \ldots, x_1^{(n)}, \ldots, x_{d_n}^{(n)}).$$

When we know what state z we want to achieve, the natural question is: can we achieve this state, i.e., are there values of the control variables which lead to this very state?

The simplest possible functional dependence is described by *linear functions*. For such functions, the question of whether we can achieve a given state z reduces to the solvability of the corresponding system of linear equations; this solvability can be checked by using known (and feasible) algorithms from linear algebra.

Next in complexity is the case when instead of a linear dependence, we have a *multi-linear* one, i.e., a dependence $F : R^{d_1} \times \ldots \times R^{d_n} \to R^d$ of the form

$$z_i = \sum_{i_1=1}^{d_1} \ldots \sum_{i_n=1}^{d_n} a_{ii_1\ldots i_n} \cdot x_{i_1}^{(1)} \cdot \ldots \cdot x_{i_n}^{(n)}.$$

Is the above problem still algorithmically solvable for such dependences? Is it feasibly solvable? These are the questions which we answer in the present paper.

In other words, we answer the following questions: Is there a (feasible) algorithm that would, given a multi-linear mapping, determine its image? Or, at least, given a multi-linear mapping and a point, determine whether this given point belongs to the image of a given multi-linear mapping?

In this paper, we will prove two things:

- first, that there do *exist algorithms* for determining the desired shape and for checking whether a given point belongs to a given image;
- second, that *no feasible algorithm is possible* for these tasks, or, more precisely, that even for bilinear mappings, this problem is computationally intractable (NP-hard).

Comment. For the exact definition of the notion of NP-hardness, see, e.g., [6]. For the readers who are not well familiar with this notion, in the following sections, we will give an explanation of what NP-hard means.

2 The Problem Is (In Principle) Algorithmically Solvable

To describe a general shape of a multi-linear mapping, we must recall the following definition (see, e.g., [1]):

Definition 1. *Let n be a positive integer, and R^n be an $n-$dimensional vector space. A set $S \subseteq R^n$ is called semi-algebraic if this set is a union of finitely many sets $S_1 \ldots, S_p$, each of which consists of all tuples that satisfy one or several conditions of the types*

$$P_j(x_1, \ldots, x_n) = Q_j(x_1, \ldots, x_n), \tag{1}$$

$$P_k(x_1, \ldots, x_n) > Q_k(x_1, \ldots, x_n), \tag{2}$$

or

$$P_l(x_1, \ldots, x_n) \geq Q_l(x_1, \ldots, x_n), \tag{3}$$

for some polynomials P_i and Q_i.

Definition 2. *A real number r is called algebraic if $P(r) = 0$ for some polynomial $P(x)$ with integer coefficients that is not identically 0.*

PROPOSITION 1.

- *The image of every multi-linear mapping is a semi-algebraic set.*
- *If all the coefficients $a_{ii_1\ldots i_n}$ of the multi-linear mapping are algebraic numbers, then each set S_k in the description of the image as a semi-algebraic set can be described by conditions $(1)-(3)$ in which all the coefficients of all the polynomials P_i and Q_i are algebraic numbers.*

Comments.

- For readers' convenience, all the proofs are given in the last section of this paper.
- Let us now describe what computability means for algebraic numbers, multi-linear mappings, and image sets.

Definition 3. *We say that an algebraic number r is given to an algorithm if this algorithm can use the following information:*

- *the integer coefficients of a polynomial $P(x)$ for which $P(r) = 0$;*
- *the rational endpoints a, b of an interval $[a, b]$ on which r is the only root of the polynomial $P(x)$.*

Comment. If we know the coefficients of the polynomial $P(x)$, and if we know the interval $[a, b]$ on which this polynomial has exactly one root r, then known numerical methods can easily compute this root r with arbitrary accuracy.

Definition 4.

- *We say that an algebraic number r is given to an algorithm if this algorithm can use the following information:*
  - *the integer coefficients of a polynomial $P(x)$ for which $P(r) = 0$;*
  - *the rational endpoints of an interval on which r is the only root of the polynomial $P(x)$.*
- *We say that a multi-linear mapping is given to an algorithm, if this algorithm can use the following information:*
  - *a positive integer d;*
  - *a positive integer n;*
  - *n positive integers $d_1, \ldots, d_n$;*
  - *$d \times d_1 \times \ldots \times d_n$ algebraic numbers $a_{ii_1 \ldots i_n}$.*
- *We say that an algorithm computes an algebraic number r if it computes two things:*
  - *the coefficients of a polynomial $P(x)$ with integer coefficients for which $P(r) = 0$;*
  - *the rational endpoints a, b of an interval $[a, b]$ on which r is the only root of the polynomial $P(x)$.*
- *We say that an algorithm computes a polynomial $P(x_1, \ldots, x_n)$ if:*
  - *this polynomial has algebraic coefficients and*
  - *the algorithm computes all the coefficients of this polynomial.*

- *We say that an algorithm computes a semi-algebraic set S if the algorithm:*
  - *produces the list of equalities and inequalities* (1) – (3) *that define the set S, and*
  - *computes all the polynomials from these equalities and inequalities.*

PROPOSITION 2. *There exists an algorithm that, given a multi-linear mapping, computes the (semi-algebraic) image of this mapping.*

Comments.

- The mere *existence* of the algorithm does not necessarily mean that this algorithm is practical. Indeed, as we will see from the proof, the algorithm that we propose uses an algorithm proposed by A. Tarski, and it is known that for the problem solved by Tarski's algorithm, for some cases, at least doubly exponential time is necessary, i.e., time $\geq 2^{2^s}$, where s is the length of the input (in bits); see, e.g., [4]. Even for small s, 2^{2^s} is unrealistically large. Thus, the running time of *our* algorithm is at least doubly exponential and hence, unrealistic. In the following text, we will show that realistic algorithms are, most probably, not possible at all.

- Since the problem of describing the image itself turns out to be too complicated, we may want to consider a simpler problem: checking whether a given point belongs to a given image. (In the following text, this easier problem will also be shown to be computationally intractable.) Let us first describe this problem formally.

Definition 5. *We say that a point $z = (z_1, \ldots, z_d) \in R^d$ is given to an algorithm if all d coordinates $z_1, \ldots, z_d$ of this point are given to this algorithm.*

PROPOSITION 3. *There exists an algorithm that, given a multi-linear mapping $f : R^{d_1} \times \ldots \times R^{d_n} \to R^d$ and a point $z \in R^d$, checks whether the given point z belongs to the image of the given mapping.*

Comment. Before we start formulating and proving our result (that finding the image of a multi-linear mapping is computationally intractable), let us briefly recall what the terms *computationally intractable* and *computationally feasible* mean.

3 Feasible and Intractable: General Definitions (Brief Reminder)

3.1 Feasible

Some algorithms require lots of time to run. For example, some algorithms require the running time of $\geq 2^s$ computational steps on an input of size (bit length) s. For reasonable sizes $s \approx 300$, the resulting running time exceeds the lifetime of the Universe and is, therefore, for all practical purposes, non-feasible.

In order to find out which algorithms are feasible and which are not, we must formalize what "feasible" means. This formalization problem has been studied in theoretical computer science; no completely satisfactory definition has yet been proposed.

The best known formalization is: an algorithm $\mathcal{U}$ is *feasible* if and only if it is *polynomial time*, i.e., if and only if there exists a polynomial P such that for every input x, the running time $t_{\mathcal{U}}(x)$ of the algorithm $\mathcal{U}$ on the input x is bounded by $P(|x|)$ (here, $|x|$ denotes the length of the input x).

This definition is not perfect, because there are algorithms that are polynomial time but that require billions of years to compute, and there are algorithms that require in a few cases exponential time but that are, in general, very practical. However, this is the best definition we have so far.

3.2 Intractable (NP-hard)

For many mathematical problems, it is not yet known (2000) whether these problems can be solved in polynomial time or not. However, it is known that some combinatorial problems are as tough as possible, in the sense that if we can solve any of these problems in polynomial time, then, crudely speaking, we can solve many practically important combinatorial problems in polynomial time. The corresponding set of important combinatorial problems is usually denoted by NP, and problems whose fast solution leads to a fast solution of all problems from the class NP are called *NP-hard*. The majority of computer scientists believe that NP-hard problems are not feasible. For that reason, NP-hard problems are also called *intractable*. For formal definitions and detailed descriptions, see, e.g., [6].

3.3 Propositional satisfiability: historically the first example of an NP-hard problem

The standard method of proving that some problem is NP-hard is by *reducing* this problem to some other problem for which NP-hardness has already been proved. Historically the first example of an NP-hard problem was the so-called *propositional satisfiability problem* for 3-CNF formulas.

This problem can be formulated as follows:

- Let $v_1, \ldots, v_k$ be a finite list of *Boolean* (*propositional*) variables, i.e., variables that take two possible values: "true" and "false".
- By a *literal*, we mean either a variable v_i, or its negation $\neg v_i$; the negation $\neg v_i$ will also be denoted by v_{-i}.
- By a *disjunction* (*clause*) D, we mean an expression of the type $a \vee \ldots \vee b$, where $a, \ldots, b$ are literals.
- By a 3-*CNF* formula, we mean an expression of the type $D_1 \& \ldots \& D_m$, where $D_1, \ldots, D_m$ are disjunctions each of which has two or three literals.
- By a *Boolean vector*, we mean a sequence of k truth values $v_1, \ldots, v_k$.
- For each Boolean vector, we can define the truth value of a CNF formula F by substituting the values v_i into the formula F.
- We say that a propositional formula is *satisfiable* if there exists a Boolean vector that makes this formula true.

  For example, a formula $(v_1 \vee v_2) \& (v_1 \vee \neg v_2)$ is satisfied by a Boolean vector $v_1 = v_2 =$"true".
- By a *SAT problem*, we mean the following problem:

  GIVEN: a 3-CNF formula;

  CHECK: whether this formula is satisfiable.

Now, we are ready to formulate our results about computational intractability.

4 The Problem Is Computationally Intractable

It turns out that our problem is computationally intractable even in the simplest case:

- when we only consider *bilinear* mappings;
- when we only consider mappings with *integer* coefficients $a_{ii_1i_2}$;
- when we are not interested in computing the entire image set, but only in checking whether a given point $z = (z_1, \ldots, z_d) \in R^d$ belongs to this set;
- when we are only interested in point z with integer coefficients z_i.

Since the problem is computationally intractable even for this simple case, an arbitrary more general problem is also computationally intractable.

PROPOSITION 4. *The following problem is computationally intractable (NP-hard):*

GIVEN:
- *a bilinear mapping $f : R^{d_1} \times R^{d_2} \to R^d$ with integer coefficients $a_{ii_1i_2}$, and*
- *a point $z = (z_1, \ldots, z_d) \in R^d$ with integer coordinates z_i,*

CHECK: whether the given point z belongs to the image of the given mapping f.

5 Proofs

5.1 Proofs of Propositions 1–3

The fact that the point $z = (z_1, \ldots, z_d)$ belongs to the image of a multi-linear mapping can be described by the following logical formula:

$$\exists x_1^{(1)} \ldots \exists x_{d_1}^{(1)} \ldots \exists x_1^{(n)} \ldots x_{d_n}^{(n)} \left(z_1 = \sum_{i_1=1}^{d_1} \ldots \sum_{i_n=1}^{d_n} a_{1i_1\ldots i_n} \cdot x_{i_1}^{(1)} \cdot \ldots \cdot x_{i_n}^{(n)} \;\&\; \ldots \;\&\; z_d = \sum_{i_1=1}^{d_1} \ldots \sum_{i_n=1}^{d_n} a_{di_1\ldots i_n} \cdot x_{i_1}^{(1)} \cdot \ldots \cdot x_{i_n}^{(n)} \right).$$

This formula belongs to a general class (called *first order formulas of real numbers theory*) described by A. Tarski in [12]. Namely, in this class:

- We start with constants 0 and 1, and with variables that run over all real numbers.
- By applying standard arithmetic operations $+, -, \cdot, :$, to these constants and variables, we get elementary expressions called *terms*.
- An *elementary formula* is an expression of one of the types $t_1 < t_2$, $t_1 = t_2$, $t_1 > t_2$, $t_1 \leq t_2$, $t_1 \neq t_2$, and $t_1 \geq t_2$.
- Finally, a *formula* is any expression that can be obtained from elementary formulas by using propositional connectives (such as $\vee$, &, $\neg$, $\to$ (*implies*), etc.), and quantifiers $\forall x$ and $\exists x$ over all real numbers.

Comments.

- For general notions of logic, see, e.g., [2, 5, 10].
- In our definition, we started with *two* real numbers 0 and 1. It is known that if we start with *arbitrary algebraic* real numbers, we end up with the same class of first order formulas. Indeed, if we know that r is the only real number on the interval $[a, b]$ for which $P(r) = 0$ for a given polynomial $P(x)$ with integer coefficients, then we can use this fact to re-formulate an arbitrary formula that contains r: e.g., the formula $r + 1 = s$ can be reformulated as

$$\forall x((a \leq x \,\&\, x \leq b \,\&\, P(x) = 0) \to x + 1 = s).$$

Tarski has shown ([12], see also [1]) that there exists an algorithm that checks, for each such formula, whether it is true or not. Thus, by applying Tarski's algorithm to the resulting formula, we will be able to check whether the given vector $z \in R^d$ belongs to the image of the desired mapping. Proposition 3 is proven.

It is also known that for an arbitrary formula from this class, the set of all vectors satisfying this formula is semi-algebraic [11, 12], and that there exists an algorithm that computes the coefficients of the corresponding polynomials P_j and Q_j. Thus, Propositions 1 and 2 are also proven.

5.2 Proof of Proposition 4

To prove this result, we will show that if we can solve our problem in polynomial time, then we will be able to solve the propositional satisfiability problem SAT for 3-CNF formulas in polynomial time. Since the problem SAT is known to be NP-hard, from this reduction, it will follow that our problem is also NP-hard.

Indeed, let us assume that we can solve our problem in polynomial time. Let us show how we can then solve any instance of SAT in polynomial time. Let $F = D_1 \& \ldots \& D_m$ be a formula in 3-CNF with variables $v_1, \ldots, v_k$. Based on this formula, we will construct a system of bilinear equations that has a solution if and only if the formula F is satisfiable. This system will have a solution if and only if a certain point with integer coefficients belongs to the image of a certain bilinear mapping with integer coefficients. The size of the data that describes this mapping will be bounded by a polynomial (actually, by the square) of the size of the original problem. Since we assumed that we can check, in polynomial time, whether a given point belongs to an image of a given mapping, we will thus be able, given a formula in 3-CNF, to check, in polynomial time, whether a given formula is satisfiable or not.

Let us now construct the desired system of bilinear equations. Let us first agree on notations:

- an integer variable i will take values from 1 to k;
- an integer variable j will take values from 1 to m; while
- integer variables α, β, and γ will take values from $-k$ to -1 and from 1 to k.

The desired mapping will be defined on the following linear spaces:

- As a first space R^{d_1}, we will take a space of dimension $d_1 = 1 + 2k + (2k)^2$. Vectors from this space R^{d_1} will be denoted by $x = (x_0, \{x_\alpha\}, \{x_{\alpha\beta}\})$; their coefficients will be denoted by x_0, x_α, and $x_{\alpha\beta}$.
- As a second space R^{d_2}, we will take a space of dimension $d_2 = 1 + 2k$. Vectors from this space R^{d_2} will be denoted by $y = (y_0, y_{-k}, \ldots, y_{-1}, y_1, \ldots, y_k)$; their coefficients will be denoted by y_0 and y_α.

Let us describe the desired bilinear equations:

- First, we construct an equation

$$x_0 \cdot y_0 = 1. \tag{4}$$

- Second, for every i from 1 to k, we construct the following two equations:

$$x_i \cdot y_0 + x_{-i} \cdot y_0 = 1; \tag{5}$$

$$x_0 \cdot y_i + x_0 \cdot y_{-i} = 1. \tag{6}$$

- Third, for every $\alpha = -k, \ldots, -1, 1, \ldots, k$, we construct an equation

$$x_\alpha \cdot y_0 - x_0 \cdot y_\alpha = 0. \tag{7}$$

- Fourth, for every α and β, we construct an equation

$$x_\alpha \cdot y_\beta - x_{\alpha\beta} \cdot y_0 = 0. \tag{8}$$

- Fifth, we add equations that correspond to m different disjunctions D_j, $1 \le j \le m$:
  - For every disjunction $D_j = v_\alpha \vee v_\beta$ that consists of two literals, we add the equation
  $$x_{\alpha\beta} \cdot y_0 = 0. \tag{9}$$
  - For every disjunction $D_j = v_\alpha \vee v_\beta \vee v_\gamma$ that consists of three literals, we add the equation
  $$x_{\alpha\beta} \cdot y_\gamma = 0. \tag{10}$$

The left-hand sides of the equations (4) – (10) form a bilinear mapping from $R^{d_1} \times R^{d_2}$ to R^d, where $d = 1 + 2k + 2k + (2k)^2 + m$; it is easy to see that all the coefficients of this bilinear mapping are integers (moreover, they only take the values -1, 0, and 1). The solvability of this system of equations is equivalent to checking whether the point $z = (1, 1, \ldots, 1, 0, \ldots, 0)$ formed by the right-hand sides of these equations belongs to the image of this bilinear mapping.

Let us show that this system of equation has a solution if and only if the original propositional formula F is satisfiable.

- Let us first assume that F is satisfiable. This means that there exists a Boolean vector $v = (v_1, \ldots, v_m)$ that makes this formula F true. Let us show that in this situation, the following x and y are the solution to the system of equations (4) – (10):

- $x_0 = y_0 = 1$;
- $x_i = y_i = 0$ if $v_i =$"true", and $x_i = y_i = 1$ if $v_i =$"false", $1 \le i \le k$;
- $x_{-i} = y_{-i} = 1 - x_i$, $1 \le i \le k$;
- $x_{\alpha\beta} = x_\alpha \cdot x_\beta$.

As a result of this choice, $y_\alpha = x_\alpha$, and $x_\alpha = 0$ if and only if $v_\alpha =$"true". Let us show that the equations $(4) - (10)$ are indeed satisfied for the resulting vectors $x \in R^{d_1}$ and $y \in R^{d_2}$:

- Equation (4) is trivially true.
- Equation (5) has the form $(x_i + x_{-i}) \cdot y_0 = 1$, and is true due to $x_i + x_{-i} = 1$ and $y_0 = 1$.
- Equation (6) follows from (5) since we have chosen $x_\alpha = y_\alpha$.
- Equation (7) follows directly from the fact that we have chosen $x_\alpha = y_\alpha$.
- Equation (8) is true due to our choice of $x_{\alpha\beta} = x_\alpha \cdot x_\beta$ and the fact that $y_0 = 1$ and $y_\beta = x_\beta$.
- Let us now show that the equations (9) and (10) are also true. Since the values v_i make the formula $F = D_1 \& \ldots \& D_m$ true, all disjunctions D_j must also be true. Let us consider two possible cases.
  * If D_j has exactly two literals, i.e., if $D_j = v_\alpha \vee v_\beta$, then one of these literals v_α and v_β must be true, and therefore, the real-valued variable corresponding to this literal must be equal to 0. Thus, either $x_\alpha = 0$ or $x_\beta = 0$. In both cases, we have $x_{\alpha\beta} = x_\alpha \cdot x_\beta = 0$, i.e., equation (9) is true.
  * If D_j has exactly three literals, i.e., if $D_j = v_\alpha \vee v_\beta \vee v_\gamma$, then one of these literals v_α, v_β, and v_γ must be true, and therefore, the real-valued variable corresponding to this literal must be equal to 0. Thus, either $x_\alpha = 0$, or $x_\beta = 0$, or $x_\gamma (= y_\gamma) = 0$. If $x_\alpha = 0$ or $x_\beta = 0$, then, similar to the previous case, we conclude that $x_{\alpha\beta} = 0$. Hence, $x_{\alpha\beta} \cdot y_\gamma = 0$, and the equation (10) is true. If $y_\gamma = 0$, then also $x_{\alpha\beta} \cdot y_\gamma = 0$. Thus, the equation (10) is true in both cases.

Thus, if the formula F is satisfiable, the system $(4)-(10)$ has a solution and therefore, the point z belongs to the image of the given bilinear mapping.

- Let us now show that if the system $(4)-(10)$ has a solution x and y, then the propositional formula F is satisfiable. Before we show this, we need to prove some properties of the solutions (x, y).

  1. $x_0 \neq 0$ and $y_0 \neq 0$.

     This property follows from the equation (4).

  2. if $x_\alpha = 0$, then $x_{-\alpha} \neq 0$.

     This property follows from the fact that (5) has the form

     $$(x_i + x_{-i}) \cdot y_0 = 1$$

     and from the already proven property $y_0 \neq 0$. Hence, $x_i + x_{-i} \neq 0$.

  3. $x_\alpha = 0$ if and only if $y_\alpha = 0$.

     This property follows from the equation (7) and from the already proven properties $x_0 \neq 0$ and $y_0 \neq 0$.

  4. $x_{\alpha\beta} = 0$ if and only if either $x_\alpha = 0$ or $x_\beta = 0$.

     If $x_\alpha = 0$ or $x_\beta = 0$, then $x_\alpha \cdot x_\beta = 0$, so from the equation (8), we conclude that $x_{\alpha\beta} \cdot y_0 = 0$. Since $y_0 \neq 0$, we conclude that $x_{\alpha\beta} = 0$. Vice versa, if $x_{\alpha\beta} = 0$, then from the equation (8), we conclude that $x_\alpha \cdot x_\beta = 0$ and therefore, that either $x_\alpha = 0$, or $x_\beta = 0$.

  Now, we are ready to show that the formula F is satisfiable. Namely, we will show that the following Boolean vector makes the formula F true: for every i, we take:

  - $v_i =$"true" if $x_i = 0$ and
  - $v_i =$"false" if $x_i \neq 0$.

  Let us show that for this choice, $x_\alpha = 0$ implies $v_\alpha =$"true". Indeed:

  - For $\alpha = i$, this directly follows from the definition of v_i.
  - For $\alpha = -i$, if $x_{-i} = 0$, then, due to Property 2, we have $x_i \neq 0$, hence, $v_i =$"false" and $v_{-i} = \neg v_i =$"true".

  To show that $F = D_1 \& \ldots \& D_m$ is true, we must show that each disjunction D_j is true. We will consider two possible cases:

- Let us first consider the case when D_j has exactly two literals, i.e., when $D_j = v_\alpha \vee v_\beta$. In this case, equation (9) is true, i.e., $x_{\alpha\beta} \cdot y_0 = 0$, Since $y_0 \neq 0$ (Property 1), we have $x_{\alpha\beta} = 0$. Hence, due to Property 4, either $x_\alpha = 0$ or $x_\beta = 0$. In the first case, $v_\alpha =$"true"; in the second case, $v_\beta =$"true". In both cases, $D_j = v_\alpha \vee v_\beta$ is true.
- Let us now consider the cases when D_j has three literals, i.e., when $D_j = v_\alpha \vee v_\beta \vee v_\gamma$. In this case, the equation (1) is true, i.e., $x_{\alpha\beta} \cdot y_\gamma = 0$. Therefore, either $x_{\alpha\beta} = 0$ or $y_\gamma = 0$.
  * In the first sub-case, due to Property 4, either $x_\alpha = 0$ or $x_\beta = 0$.
  * In the second sub-case, due to Property 3, $x_\gamma = 0$.

  In both sub-cases, either $x_\alpha = 0$, or $x_\beta = 0$, or $x_\gamma = 0$. Thus, either $v_\alpha =$"true", or $v_\beta =$"true", or $v_\gamma =$"true". In all three cases, $D_j = v_\alpha \vee v_\beta \vee v_\gamma$ is true.

So, the original formula F is indeed true.

Reduction to satisfiability is proven: the formula F is satisfiable if and only if the system (4) – (10) has a solution, i.e., if and only if the corresponding point z belongs to the image of the corresponding bilinear mapping. Hence, the proposition is proven.

Acknowledgments. This work was supported in part by NASA under cooperative agreement NCC5-209, by NSF grants No. DUE-9750858 and CDA-9522207, by United Space Alliance, grant No. NAS 9-20000 (PWO C0C67713A6), by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-95-1-0518, and by the National Security Agency under Grant No. MDA904-98-1-0561.

We are thankful to Piotr Wojciechowski for the formulation of the problem and for valuable discussions.

References

[1] V. I. Arnold, *Geometrical methods in the theory of ordinary differential equations*, Springer-Verlag, N.Y., 1983.

[2] J. Barwise (ed.), *Handbook of Mathematical Logic*, North-Holland, Amsterdam, 1977.

[3] S. Bernau and P. J. Wojciechowski, "Images of bilinear mappings into R^3", *Proceedings of the American Mathematical Society*, 1996, Vol. 124, No. 12, pp. 3605–3612.

[4] J. H. Davenport, J. Heintz, "Real quantifier elimination is doubly exponential", *Journal of Symbolic Computations*, 1988, Vol. 5, No. 1/2, pp. 29–35.

[5] H. B. Enderton. *A mathematical introduction to logic.* Academic Press, N.Y., 1972.

[6] M. Garey and D. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.

[7] L. R. Lewis and C. H. Papadimitriou, *Elements of the theory of computation*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[8] J. C. Martin, *Introduction to languages and the theory of computation*, McGraw-Hill, N.Y., 1991.

[9] C. H. Papadimitriou, *Computational Complexity*, Addison Wesley, San Diego, 1994.

[10] J. R. Schoenfield. *Mathematical logic.* Addison-Wesley, 1967.

[11] A. Seidenberg, "A new decision method for elementary algebra", *Annals of Math.*, 1954, Vol. 60, pp. 365–374.

[12] A. Tarski, *A decision method for elementary algebra and geometry*, 2nd ed., Berkeley and Los Angeles, 1951.