Novi Sad J. Math.
Vol. 31, No. 1, 2001, 167-173

IDENTITIES IN THE POLYNOMIAL EXPRESSIONS OF MATHEMATICAL MODELS OF ROBOTIC MECHANISMS

Miloš Racković [1]

Abstract. Automatic generating of the symbolic mathematical models of robotic mechanisms belongs to the class of problems of combinative optimisation. One way of dealing with this problem is the application of object-oriented approach to design of the system for mathematical modelling of robotic mechanisms. This paper presents formal specification of the system for reducing the numerical complexity of the polynomial expressions forming the robotic mechanism model, using the Unified Modelling Language.

AMS Mathematics Subject Classification (1991): 68Q40, 70B15

Key words and phrases: object-oriented specification, trigonometric identities

1. Introduction

Considerable progress in modelling robotic mechanisms has been achieved by introducing the numeric-symbolic [1] and symbolic methods which develop special data structures for representing analytic expressions of the model and enable reduction of numerical complexity of the generated model. In [2, 3], a database management system was introduced into the modelling process of robotic mechanisms and the basic Newton-Euler method for forming the model of simple kinematic chain dynamics in closed form [1] has been broadened in such a way as to enable modelling of both complex and closed kinematic chains using the notations introduced in [4]. Now, the model of the robotic mechanism in closed form [2, 3] can be expressed as:

$$P = H(q,\Theta)\ddot{q} + \dot{q}^{\top} C(q,\Theta)\dot{q} + +h^G(q,\Theta) + B(q,\Theta)\sigma \tag{1}$$

where:

$H = H(q,\Theta)$ - inertial matrix of the mechanism;

$C = C(q,\Theta)$ - matrix of the Coriolis and centrifugal effects;

$h^G = h^G(q,\Theta)$ - gravity vector;

$B = B(q,\Theta)$ - matrix of chain closure;

σ - reaction vector of chain closure.

[1]University of Novi Sad, Faculty of Science, Institute of Mathematics, Trg D. Obradovića 4, 21000 Novi Sad, Yugoslavia

A detailed mathematical analysis shows that all model expressions are of the following form:

$$Y = \sum_{i=1}^{N} k_i \cdot \prod_{j=1}^{L} x_j^{e_{ij}} \tag{2}$$

where:
Y - robotic quantity to be calculated;

k_i - constant coefficient related to the i-th addend;

x_j - one of the variables of the robotic mechanism model;

e_{ij} - exponent of the j-th variable of the i-th addend.

It is important to mention that the same form (Eq. 2) represents both the basic mathematical expressions for forming the dynamic mechanism model and the fully-developed analytic expressions of the model. The difference is in the fact that x_j in the case of the developed expressions must be one of the basic robotic quantities (q, $\dot{q}$, $\ddot{q}$, $\sin q$, $\cos q$).

In [5], the object-oriented approach is introduced to design a system for symbolic modelling of robotic mechanisms. The formal specification of the system is given in Unified Modelling Language (UML) [6-8] by use case and class diagrams. In [9], the state diagram is given which describes the dynamics of the system only on the highest level. More details about the process for forming the starting model of the mechanism are given in [10]. This paper describes the system for reducing numerical complexity of the generated model. The main task is to specify the elimination of the identities from the fully-developed analytical expressions of the generated model.

2. Class diagram

The system statics is represented by the class diagram in Fig. 1. The basic classes of the system are **Calculating graph** and **Analytic expressions**. The class **Calculating graph** represents mathematical expressions which form the mathematical model of the mechanism dynamics, and the class **Analytic expressions** represents the fully-developed analytic expressions of the model. Both classes have the attribute *phase* which contains the integer value indicating the phase in the process of modelling.

In order to represent the polynomial expressions (Eq. 2) we introduced the abstract classes **Variable** and **Addends**. The class **Variable** is used for representing all variables in polynomial expressions which stand for robotic quantities. Each variable has the attributes *code* and *value*. The attribute *code* uniquely determines the variable and the attribute *value* stores the numerical value of the robotic quantity which is represented by that variable. The class **Addends** serves for connecting the expression with its addends. This class contains the same attributes. The attribute *code* stores the code of the variable from the left-hand side of the equality sign in the expression, and the attribute *value* stores the value of the constant coefficient of the addend.

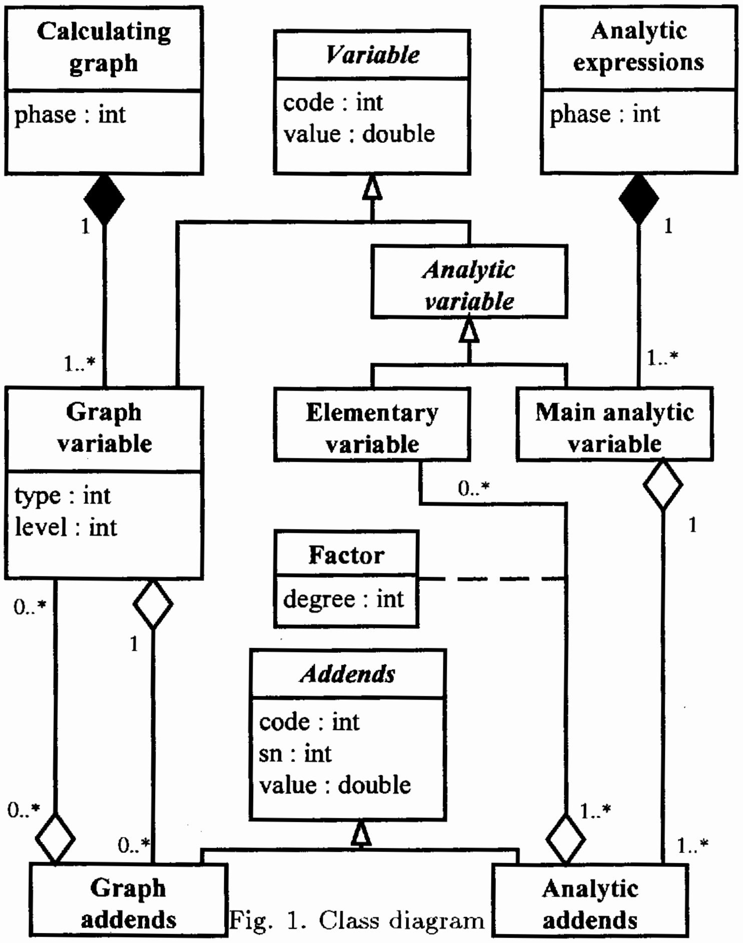

Fig. 1. Class diagram

Although both expressions are of the same type we have to introduce different classes for their representation in order to distinguish them in the process of forming the model and reducing its calculating complexity. We specialise the class **Variable** on the classes **Graph variable** and **Analytic variable**. The class **Addends** is specialised on the classes **Graph addends** and **Analytic addends** for the same reason of distinguishing the two kinds of expressions. The class **Analytic variable** does not contain any additional attributes apart from the inherited ones. On the contrary, the class **Graph variable**, in addition to the inherited attributes, have the two additional attributes *type* and *level*. The attribute *type* takes values from the set {'0','1','2','3'}, and denotes the type of the robotic quantity which is represented by the variable. The attribute *level* is introduced to enable calculating of the robotic quantities by bottom-up navigating through the database. The class **Analytic variable** is the abstract class which is also specialised on the classes **Elementary variable** and **Main analytic variable**, because the fully-developed analytic expressions contain only two kinds of variables.

Let us describe storing of the fully-developed analytic expressions. The variable from the left-hand side of the equality sign is stored as an instance of the class **Main analytic variable**. This variable (expression) is connected to its addends by aggregation to the class **Analytic addends**, where the constant coefficients of every addend are stored (in the attribute *value*) so that each addend takes a new instance of the class. Connection of the addend and its factors is represented by the aggregation from the class **Analytic addends** to the class **Elementary variable** in which all basic robotic quantities are stored. Each addend is connected only to its factors, and for every connection we have attached the association class **Factor**, which represents the degree of that factor stored in the attribute *degree*. Storing of the expressions in the form of calculating graph is analogous to the previous one but we will not describe this here because the process for eliminating the identities from the developed analytical expressions does not use these expressions.

3. Elimination of identities

When the starting model of the robotic mechanism is formed ([10]), the next step is the elimination of the redundant mathematical operations. First, we need to form the fully-developed analytic expressions and then to apply all possible trigonometric identities. The sequence diagram given in Fig. 2 specifies the process of elimination of the trigonometric identities from the fully-developed analytic expressions.

The actor *User* initialises the process by calling the method *trig_ident*. This method belongs to the class **Analytic expressions** and it is called over its instance which contains all fully-developed analytic expressions. The control is then passed to the class **Utility** (method *trig_identities*). This class contains utility methods and does not serve for storing the analytic expressions. For this reason this class is not included in the class diagram in Fig. 1. In order to apply all possible trigonometric identities we need to process all fully-developed analytic expressions, one by one. This is performed by the external loop in which we call the method *next_var_ml* belonging to the class **Analytic expressions**. This method finds the next unprocessed analytic expression, while such expression exists.

For each expression, we have to sort first its addends according to the number of factors in every addend and initialise the serial number of the addend (methods *sort_add* and *init_sn* which belong to the class **Utility**). Then, using the double loop, we pass through the addends of the current expression, so that every two addends are compared with the aim of discovering and applying the identities. Here we use the method *next_add_sn* which belongs to the class **Main**

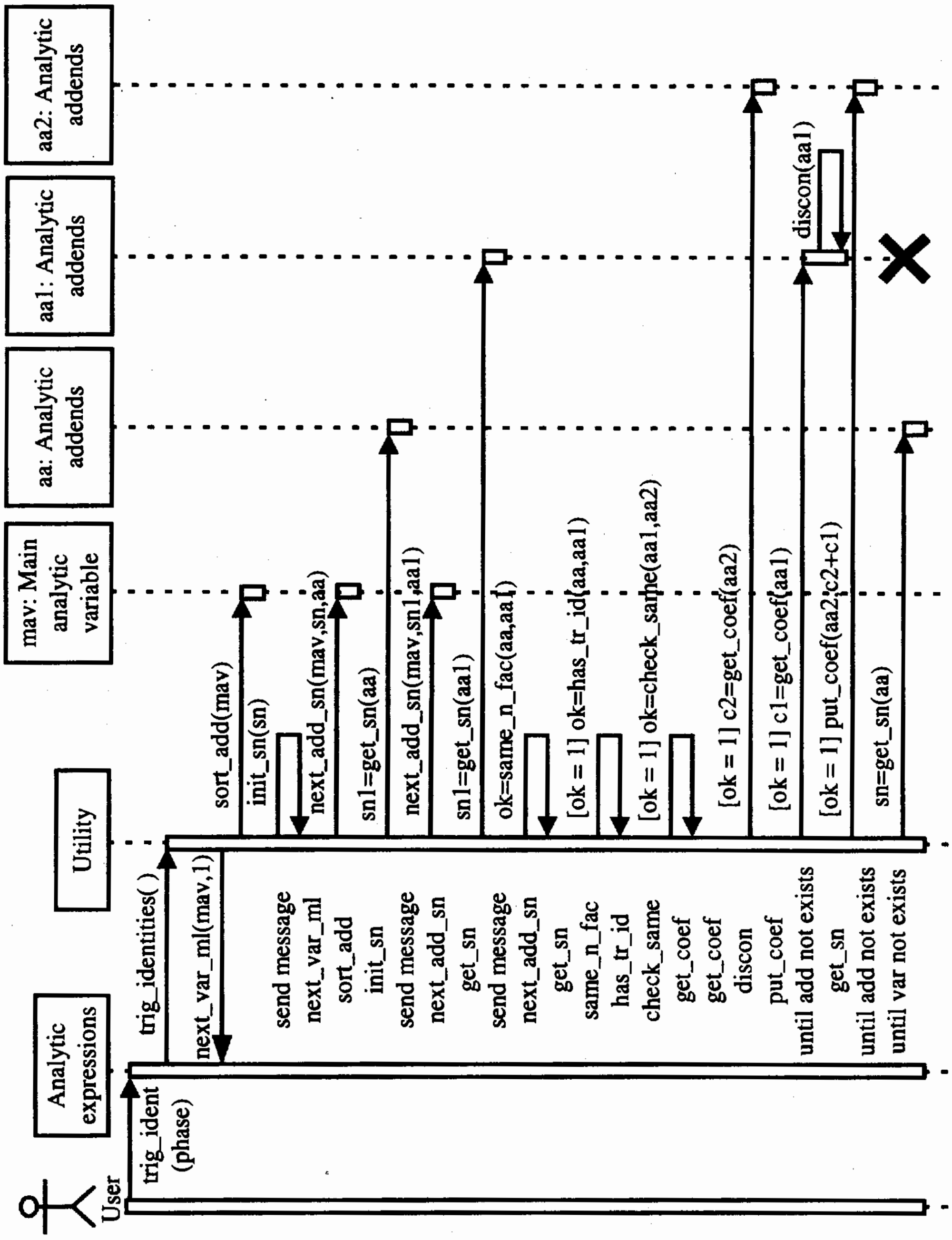

Fig. 2. Sequence diagram

analytic variable and finds the next addend according to the one with serial number *sn* which is generated for every addend by calling the method *get_sn*, belonging to the class **Analytic addends**. If we want to apply some trigonometric identity on the two addends, they must first fulfil the condition of having the same number of factors. The condition is checked by the method *same_n_fac* which belongs to the class **Utility**, and then, in the case when it is fulfilled, the method *has_tr_id* is called, belonging to the same class. This method finds the trigonometric identity which can be applied, if such identity exists, and then performs its elimination. It is possible to apply the trigonometric identities of the form:

$$\sin^2 x + \cos^2 x = 1$$

$$\sin(x+y) = \sin x \cdot \cos y + \cos x \cdot \sin y$$

$$\cos(x+y) = \cos x \cdot \cos y - \sin x \cdot \sin y$$

where x and y are either some of the internal coordinates of the robotic mechanism or the variables obtained from the mentioned identities.

In all three cases, one of the addends is eliminated and in the other one, the number of factors is reduced by one. All these operations are performed in the method *has_tr_id.* Now, the addend with the reduced number of factors can be the same as the one of the remaining addends of the current expression. If it is so, we need to unite these two addends, by summing their constant coefficients. These operations are realised by the method *check_same* which belongs to the class **Utility** and the methods *get_coef*, *put_coef* and *discon*, belonging to the class **Analytic addends**.

The described operations specify the process of eliminating the trigonometric identities. In a similar way, we can specify the process for eliminating the identical expressions too, which is also part of the process for reducing numerical complexity of the analytic expressions. The next task is to store the reduced fully-developed analytic expressions in the form of calculating graph again, which is not a subject of this paper.

4. Conclusion

By a detailed analysis of the Newton-Euler method it was established that the mathematical relations forming the robotic mechanism model are in the polynomial form. With the aim of forming mathematical models of robotic mechanisms in symbolic form, it is necessary to develop special data structures, suitable for storing the polynomial expressions and for reducing their calculating complexity.

The object-oriented approach was used to design a system for mathematical modelling of robotic mechanisms in symbolic form. Using UML, the complete formal specification of the informational requests of complex technical systems can be performed. In the class diagram, consisting part of this modelling language, the statics of the given system is described. The system dynamics can be specified too, using the use case and other UML diagrams. Elimination of the trigonometric identities from the fully-developed analytical expressions of the mechanism model is specified here by the sequence diagram. By adding specification for the other system processes we can obtain a complete formal specification of the given system, which is the basis for its implementation.

References

[1] Vukobratović, M., Kirćanski, N., *Real-Time Dynamics of Manipulation Robots* (Springer-Verlag 1985).

[2] Racković, M., Vukobratović, M., Surla, D., "Generation of Dynamic Models of Complex Robotic Mechanisms in Symbolic Form", *Robotica* (1998) volume 16, pp. 23-36.

[3] Racković, M., Surla, D., Vukobratović M., "On Reducing Numerical Complexity of Complex Robot Dynamics", *Journal of Intelligent and Robotic Systems* **24** (1999), pp. 269-293.

[4] Vukobratović, M., Borovac, B., Surla, D., Stokić, D., *Biped Locomotion, Dynamics, Stability, Control and Application* (Springer-Verlag 1990).

[5] Racković, M., Surla, D., "Object-Oriented Specification of the System for Generating Mathematical Models of Robotic Mechanisms Dynamics", *Fourth ECPD International Conference on Advanced Robotics, Intelligent Automation and Active Systems*, Moscow, 1998, pp. 270-275.

[6] Booch, G., Rumbaugh J., Jacobson, I., *The Unified Modeling Language User Guide*, Addison-Wesley Longman, Inc., 1999.

[7] Rumbaugh J., Jacobson, I., Booch, G., *The Unified Modeling Language Reference Manual*, Addison-Wesley Longman, Inc., 1999.

[8] Stanojević, I., Surla, D., *Introduction to Unified Modelling Language*, Group of Information Technologies, Novi Sad, 1999.

[9] Surla, D., Racković, M., "Object-Oriented Specification of the System for Generating Symbolic Models of Robotic Mechanisms", *Novi Sad J. Math.* Vol. 30, No. 3, 2000, pp. 149-160.

[10] Racković, M., "Specification of the Classes for Forming the Robotic Mechanism Model, *Informal Proceedings of the TARA 2000 Conference*, Novi Sad, Yugoslavia, September 6-7, 2000, pp. 85-92.