USING COLORED PETRI NETS TO MODEL AND SIMULATE CONTROL ARCHITECTURES OF MOBILE ROBOTS

Hilano José Rocha de Carvalho
School of Engineering of São Paulo, University of São Paulo
Trabalhador São-carlense Avenue, 500, CEP 13566-590
hilanorc@sc.usp.br

Rafael Vieira Sousa
School of Engineering of São Paulo, University of São Paulo
Trabalhador São-carlense Avenue, 500, CEP 13566-590
rafael@cnpdia.embrapa.br

Arthur José Vieira Porto
School of Engineering of São Paulo, University of São Paulo
Trabalhador São-carlense Avenue, 500, CEP 13566-590
ajvporto@sc.usp.br

Ricardo Yasshussi Inamasu
Embrapa Agricultural Instrumentation
XV de Novembro St., 1452, CEP 13561-160
ricardo@cnpdia.embrapa.br

Abstract. *The research on autonomous systems has been influencing the improvement of the navigation of vehicles and mobile robots using artificial intelligence techniques. Several economical segments can get advantage from the application of that scientific effort. On one hand, those systems demand control architectures which may have a high level of complexity when being tested by physical implementations. On the other hand, recent bibliography presents the efficacy of Petri nets on modeling of real systems with concurrent activities and for the planning and control of mobile robots´ tasks. The objective of this paper is to present the use of colored Petri net (CPN) formalism to model and simulate a control system for a mobile robot. A fuzzy logic behavior-based navigational control architecture CPN model is presented. The CPNs fitted well to the purpose of this work due to three main aspects: the capability of quantitative analysis of input data to the modeled robotic system, the possibility of using computational programming resources and the efficient analysis of the simulations from specific output files. Finally, a verified mathematical model of the robotic control system based on Petri nets formalism was obtained without the necessity of defining a new Petri nets´ extension.*

Keywords*: control architectures, mobile robots, colored Petri nets.*

1. INTRODUCTION

Recently, the research in robotics has become a crucial aspect for social technological development. Along the last decades, industry has been the most common economic segment in which the application of robotic system has been successful. This economical-oriented association between high technology and market demands results, in general, in cost reduction and production efficiency increase. Nevertheless, more recently, other economical segments have also been affected by the research in robotics: agriculture.

The construction of mechanical platforms for the implementation of robotic navigation technologies and data acquisition in precision agriculture motivates the development of autonomous vehicles. This is an objective orientation due to the attempt to answer a global agricultural competitiveness.

Particularly, the use of mobile robots which can be able to work in non-conventional environments – e.g. agricultural environment – has a potential aim of helping agronomical research. That is the case of the Autonomous Agricultural Vehicle (VAA – abbreviation in Portuguese) project which has the objective of the development of a mobile robot for agricultural purposes (RAM – abbreviation in Portuguese) (Porto *et al.*, 2005).

The main characteristic of the mobile robot herein studied is its autonomous property. This way, control architectures for robot navigation are demanded in order to make it possible as much independent and precise as possible.

Sousa *et al.* (2005) developed a hybrid behavior based control architecture for RAM navigation using fuzzy logic. It was successfully tested in a mini-robot platform *Khepera* (K-Team, 2005). Nevertheless, the obtaining of a mathematical model which represents the internal and external interactions that may influence the mobile robot decision-making process would permit the formal analysis of a robotic control system. Moreover, being capable of foreseeing the robotic control system functioning by modeling and simulation techniques may facilitate the choice of a set of the most adequate behaviors in accordance with determined conditions and may pave the way for the optimization of robotic tasks.

The issue of robotic task control, specifically once considering unstructured environment where variability is quite common, demands a modeling tool which permits a fast visualization of the involved processes, mainly when parallel activities are present. Moreover, another aspect of the desired computational tool is related to the necessity of the verification and analysis of the constructed models. Effectively, this last point is concerned with the evaluation of different techniques and procedures, and by consequence, the proposal of new methodologies.

Among a vast set of computation techniques available, the Petri nets possess all the demanded potentialities described above with a solid mathematical formalism and proper graphical representation (Murata, 1989). This way, the Petri nets´ formalism permits the modeling and simulation and the formal analysis of different types of systems in an integrated manner. For the modeling of robotic control systems, where simultaneity of events due to parallel activities is recurrent, the Petri nets possess intrinsic characters to deal with it. In addition, the resultant control system Petri net model may be directly applied to interact with virtual or even real environments.

Comparing to other common techniques for robotic control system, such as procedural and object-oriented programming languages or some others specifically defined for the concerned purposes, none of them presents an explicit mathematical formalism like Petri nets do. Nevertheless, the inner contradictions revealed by the drawbacks (explosion of modeling elements) of the modeling of large and complex systems have been one of the motivations of the dialectical development of Petri nets´ formalism, which culminated with the definition of the so-called Petri nets´ extensions. The great resultant diversity has produced the high level Petri nets which lead the current synthesis. Among the different types of high level Petri nets, the colored Petri nets´ formalism as defined by Jensen (1997) was herein used with the aim of studying its viability as a modeling and simulation language for a robotic control system, especially due to the facilitation of editing, simulating and state space analysis provided by the CPN tools 2.2.0 (CPNTools, 2006).

This remainder of this paper consists of five subsequent sections. In section II, a brief bibliographic review is presented. Section III presents the navigational architecture proposed for RAM and tested in a mini-robot platform. Section IV presents the formal definitions of colored Petri nets (CPN) and some of the main characters of CPN tools 2.2.0., used for the modeling and verification of the Navigational Control Architecture Model based on CPN formalism (NCAM-CPN). The NCAM-CPN is described in section V and verified in section VI. Finally, conclusions and further works are discussed in Section VII.

2. BIBLIOGRAPHIC REVIEW AND RELATED WORKS

In order to provide a reasonable mechanism for a mobile robot to work as independent and accurate as possible, control architectures for navigation are necessary. A seminal paper concerning the development of alternative techniques to classical control theory for mobile robots is the result of the work of Brooks (1986). In this paper, Brooks (1986) proposes a multi-layered system composed by integrated modules with specific functions, the subsumption architecture. Each module is actually an extended finite state machine with instance variables capable of associating itself with LISP data structure. From this work on, the development of the robotic behaviors for robot control was initiated.

Among the different architectures that are present in the accumulated bibliography of decades of research, those derived from the concepts of the hybrid deliberative/reactive paradigm are in the vanguard of the studies (Murphy, 2000; Arkin, 2000). That two-layer paradigm consolidates the idea of behavior-based robots. A deliberative layer is responsible for planning the general task and subdividing it into subtasks. Besides, associated with the subtasks established, there are the possible behaviors to certain situations and environmental conditions. Based on the combined behaviors activated, a reactive layer is in charge of implementing which action is the most appropriate due to all the physical factors involved. Finally, the motor commands, which determine the motion of a robot, depend on that resulting behavior determination.

The applications of Petri nets in modeling and control of robotic tasks have been one of the foci of academic research through the last decades. This fact is easily perceived both in industrial robotics, for instance, with specific implementations in Flexible Manufacturing Systems (FMS) (Desrochers and Al-jaar, 1995).

Wang *et al*. (1991) develop a three-level structure for what he calls intelligent machines using Petri nets: organization, coordination and execution level. The organization level defines the necessary general procedures to a robotic motion, when a certain task is to be concluded. The coordination level works as a connection between the organization and the execution level, being subdivided into a dispatcher and several coordinators. The dispatcher collects the general task plans from the organization level, decompose them into control actions and distribute them to the correspondent coordinators with qualitative requirements. The coordinators, in turn, translate those control commands to operational instructions and, finally, send them to the appropriate devices for execution. The execution level carries out the instructions sending to the coordination level reports of the obtained results.

In another paper, Wang and Saridis (1993), alongside the concepts of the hierarchical levels and the definitions of cost functions and reliability measures, define Petri nets transducers (PNTs). This extension to Petri nets may work as a basic module for an analytical model, providing a formal description for individual processes in relation to the dispatcher and the coordinators. The PNTs are efficient for modeling situations that may involve concurrency and conflict for control and synchronization of operations.

Lima and Saridis (1996) define a method to obtain the robotic task optimization. Several general tasks are defined in the organization level based on a hierarchical model, which is subdivided into levels that are capable of interacting with each other constantly. Those general tasks may be alternatives for the completion of a certain pre-defined aim, which is based on a pre-defined sequence of primitive tasks. These primitive tasks, in turn, are associated with the cost functions (*J*). Besides, in a coordination level, primitive actions for each primitive task are also defined with the association of cost functions values (*J*). According to Lima and Saridis (1996), minimizing the value for *J* will result in the choice of the optimal task.

Lima *et al*. (1998) go further and consolidate the concepts about robotic task theory with a set of several primitive tasks that are internally defined by primitive actions. The authors present results of successful empirical implementations, illustrating examples of models in Petri nets. Milutinovic and Lima (2002) also implement this hierarchy between primitive tasks and primitive actions, obtaining satisfactory practical results.

Caloini *et al*. (1998), in turn, present a new approach based on control nets. A control net is defined as a high level Petri net, specifically predicate-transition Petri net, applied to control systems validation during design stage. Four basic elements are defined for the control nets: events, states, data and parallelism. The control nets are implemented with fixed blocks that are used for model development. First of all, one has to determine specific functions for modeling. A case study validates preliminary implications.

Similarly to the work by Caloini *et al*. (1998), Montano *et al*. (2000) use one of the interpreted Petri nets extensions: the Time Petri Nets (TPNs). The transition may fire or not in a determined interval of time which is established with a minimum value (X) and a maximum vale (Y), where Y is greater or equal to X. Using TPNs, one may be capable of modeling the occurrence of time-outs, periodical activities, as well as synchronization and concurrency better than Timed Petri Nets, in which the transition firing actually occurs after a pre-defined and fixed time value.

Considering the concepts implemented in the papers discussed above, the application of Petri nets, in the context of the development of autonomous robots for unstructured environments, such as mobile robots for agricultural purposes, motivated the work of Carvalho *et al.* (2005). As a matter of fact, the herein paper is a continuity of the latter´s attempt to formalize a robotic architecture by means of Petri nets. The main difference between both works is that Carvalho *et al.* (2005) constructed a Petri net model only capable of modeling the simultaneity of events in a qualitative manner, that is, the effective decision-making process was not possible by means of a low-level Petri net lacking any other type of computation support. Herein, the initial results of the cited work were absorbed with the intention to increase its capability for both qualitative and quantitative aspects by means of using a high level Petri nets´ approach.

3. A ROBOTIC ARCHITECTURE FOR NAVIGATIONAL PURPOSES OF A MINI-ROBOT

One of the current problems of mobile robots' navigation deals with the applications of the hybrid deliberative reactive architectures in outdoor environments. The conditions under which a robot is subjected may change very often. Consequently, the level of uncertainty increases. Therefore, fuzzy logic has been applied to behavior-based robots giving them the capacity to handle unusual situations in decision-making process (Seraji and Howard, 2002).

Sousa et al. (2005) developed a hybrid behavior based control architecture for RAM (agricultural mobile robot) navigation using fuzzy logic. All the field tests were carried out in a mini-robot platform, the commercial robot *Khepera* (K-Team, 2006). Briefly, it consists of a Khepera basic module, Khepera IO turret and a perceptual circuit mounted on the IO turret based on VT935G LDR sensors. The LDR sensors are located in front of the bottom circuit board of the base module to read the reflect light by the floor following or looking for a path (dark line). There are also six front infrared (IR) sensors of the basic module, which are grouped in three pairs of adjacent sensors composing three perception areas: front, left and right. The IR sensors, in turn, are in charge of detecting obstacles. Figure 1 shows the Khepera-based platform.

From the information captured by the LDR sensors, left and right, two crisp values are defined: the DIST and DIF values. The DIST crisp value determines how distant the LDR pair is becoming out of the path (intensity) – the minimum value between the LDR_L and LDR_R – and the DIF crisp value denotes which sensor is more distant from the path (direction). The crisp inputs for follow path behavior are composed of three fuzzy terms: far (FAR), medium (MEDIUM) and close (CLOSE) for DIST inputs, and negative (NEG), zero (Z) and positive (POS) for DIF inputs. The behavior outputs are composed of two values obtained by the centroid defuzzyfication method. Both outputs values are referred to each motor, left (L) or right (R). Three fuzzy terms are applied to describing the outputs: forward (F), stop (S) and backward (B).

A similar approach using the two crisp values, DIST and DIF, is used for the avoid obstacle behavior procedure. However, the DIST value indicates the distance from a detected object and which group is the closest from the object (intensity and direction) – the maximum value between the left and right IR pair grouped (G) – and the DIF value denotes which sensor of the group is closer from the object, that is, the free and occupied spaces (direction).

The fuzzy behavior arbitration, in turn, defines a hierarchy where the avoid obstacle behavior has the highest priority, follow path behavior has an intermediary priority and the straight in line behavior has the lowest priority. This approach depends on the IR or LDR sensor data about obstacle and path detections.

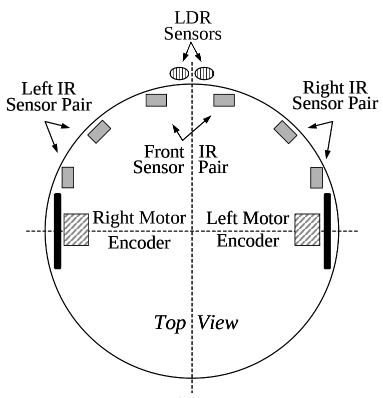


Figure 1. A top view of Khepera-based platform.

4. COLORED PETRI NETS

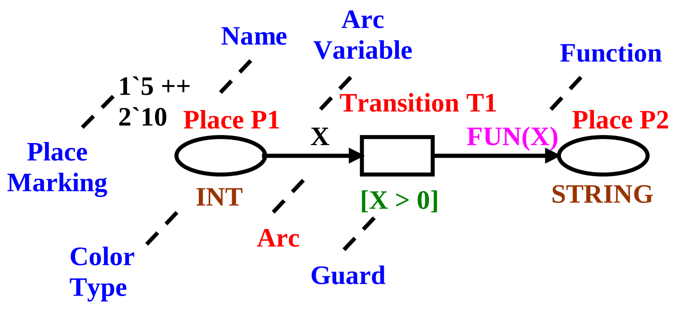
The theory of colored Petri nets (CPN) was developed as an extension to the basic Petri nets` theory (Murata, 1989). The prime objective of CPN is to make feasible the modeling and formal analysis of large, concurrent and distributed systems using Petri nets. A detailed definition and discusson about the CPN formalism can be found in Jensen (1997).

Succinctly, a simple CPN model is defined by Desrochers and Al-jaar (1995) as a bipartite directed graph represented by a quintuple $CPN = (P,T,C,I,O)$, where:

- $P = \{p_1,\ldots,p_i\}$ is a finite set of places;
- $T = \{t_1,\ldots,t_i\}$ is a finite set of transitions;
- C are the set of colors associated with the places (P) and the transitions (T) such that (considering *as* and *bs* the associated colors):
  - $C(p_i) = \{a_{i1},\ldots,a_{iui}\},\ u_i = |C(p_i)|,\ i = 1,\ldots,n;$
  - $C(t_j) = \{b_{j1},\ldots,b_{jvj}\},\ v_j = |C(t_j)|,\ i = 1,\ldots,m;$
- I is an input mapping $C(p)\ x\ C(t) \rightarrow N$ (nonnegative integers) corresponding to the set of colored directed arcs from P to T;
- O is an output mapping $C(t)\ x\ C(p) \rightarrow N$ (nonnegative integers) corresponding to the set of colored directed arcs from T to P.

The CPN tools as defined in CPNTools (2006), in turn, comprise two components: a graphical user interface (GUI) and CPN ML. These ones are directly related to three integrated tools: the CPN editor, the CPN simulator and the CPN state space tool. The GUI and the CPN ML work as separate processes communicating by TCP/IP. The CPN ML is a functional programming language implemented on top of a SML/NJ compiler. Herein used, the CPN tools 2.2.0 academic version was obtained from CPNTools (2006).

Based on the CPN formalism and the CPN tools as defined above, the token primitive and compound types are defined by the color sets associated with each place and their type modifications are accomplished by functions attached to the arcs. In order to do that, the correspondent transition must be enabled to fire. This happens when the number and the type of the input place tokens to the transition agree with the conditions of transition's enablement defined by the transition guard. Figure 2 shows the CPN basic modeling elements, their graphical representation and an example of how to deal with the CPN formalism.



```
colset INT = integer;
colset STRING = string;
val X : INT;
FUN(X) =
If (X >= 5) andalso (X < 10) then 1`¨YES¨
                    else
1`¨NO¨;
```

Figure 2. CPN basic modeling elements, their graphical representation and a CPN model example with its CPN ML code.

As can be seen in Figure 2, the initial state of the CPN model is represented by three tokens in “Place P1”. The number and the type of those tokens are defined by the initial place marking of P1 (in this case, a five integer token and two ten integer tokens). They are three true firing conditions to “Transition T1” since they are in agreement with the transition guard. The firing of T1 initializes the function “FUN(X)”. “FUN(X)” captures the token data by the arc variable “X” and transforms them into one string token in number and in type following the “if-then-else” structure of the CPN ML code of Figure 1. Finally, “Place P2” receives the one string token from “FUN(X)”.

The prior conceptual discussion and the example depicted in Figure 2 are actually restricted to a non-hierarchical colored Petri net.

Herein, a hierarchical colored Petri net is proposed which may incorporate two fundamental elements of modularization:

- Substitution transitions – permits the derivation of simpler submodels by others already defined using a hierarchy approach of pages;
- Fusion places – permits a faster mechanism of distribution of token data along the entire model linking submodels that may not be hierarchically associated.

The colored Petri nets’ basic modeling elements can also be classified into static and dynamic elements. The static elements are the places, the transitions and the arcs. A node comprised of a place and a transition linked by an arc corresponds to the basic structure of any extension to Petri nets. The dynamic element that goes through the different constituent nodes of a model is the token which may suffer several types of modifications in number or in type in colored Petri nets.

In practice, the colored Petri nets alongside CPN Tools possess a computational power similar to conventional programming languages, such as C, Pascal and so on, and the simulation tools like, for example, ARENA®. This mutant character is essential for a quasi-complete analysis of systems of any kind. However, theoretical discussions concerning this subject are out of the scope of this paper.

5. MODELING OF A ROBOTIC ARCHITECTURE USING COLORED PETRI NETS: NCAM-CPN

The modeling of the robotic control architecture as defined in Section III was based on the colored Petri nets´ definitions of Section IV, using the appropriate mechanisms of modularization (substitution transitions and fusion places) in order to reach a hierarchical colored Petri nets (HCPN). The dynamic character of the token, as mentioned in Section IV, was crucial for the modeling implemented. As a matter of fact, the latter may be associated with the state of an object, the value of a variable or any kind of content that might be relevant at the modeling stage. Indeed, a composition of the static and the dynamic elements may assume functional forms in a dynamic manner. Both, the static and the dynamic elements, however, remain essentially the same, being changeable according to necessity, not assuming a definite or rigid form. These mutant characters confer to the CPN formalism the necessary flexibility to provide appropriate simulation models for multi-level of abstraction systems.

This way, the navigational control architecture model based on colored Petri nets (NCAM-CPN) is an HCPN by definition and is summarized in Figure 3.

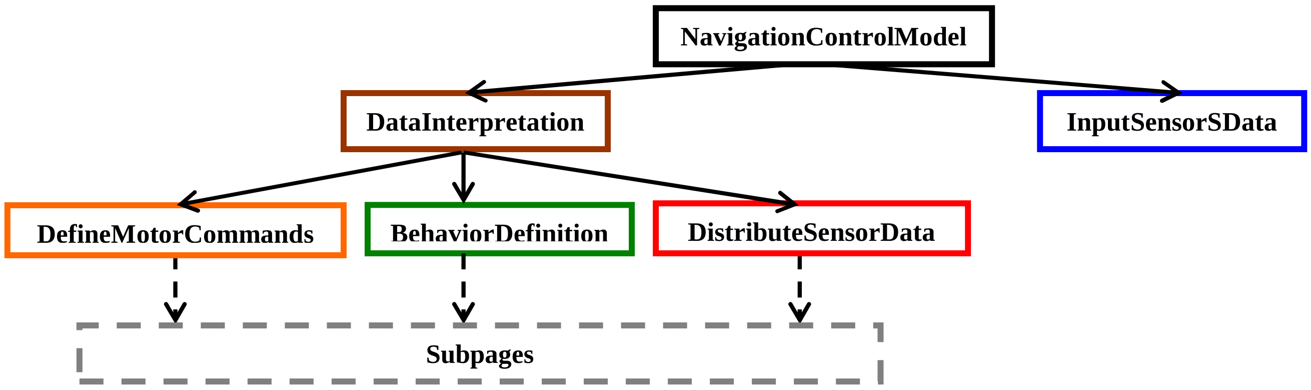


Figure 3. The NCAM-CPN prime page and its subpage derivations.

As can be seen in Figure 3, NCAM-CPN is divided into different levels of hierarchy pages. Each level of hierarchy pages means actually different levels of Petri net submodels which are linked by functionality. In order to facilitate the modeling of such a complex and large robotic control system herein studied, specialized submodels were defined in a functional fashion.

According to the theory of Petri nets, this multifunctional hierarchical model organization is implemented by the derivation of subpages from superpages (Jensen, 1997). In Figure 3, the “NavigationalControlModel” page is called the prime page of NCAM-CPN. It is considered the superpage of all the other derived subpages and so is the highest

NCAM-CPN level of abstraction. It works as an interface between the programmer or the modeler and the model in itself. From the “NavigationalControlModel” prime page, the “InputSensorSData” and the “DataInterpretation” subpages derive.

The “InputSensorSData” subpage works as the effective medium by which the sensor data are inputted to the NCAM-CPN. Subsequently, the sensor data input is passed to the “DataInterpretation” subpage. From the “DataInterpration” subpage three other subpages are derived: the “DistributedSensorData”, the “BehaviorDefinition” and the “DefineMotorCommands”.

The sensor data provided by the “InputSensorSData” subpage are collected by the “DistributedSensorData” subpage. By the latter, the quantitative aspect is initially accomplished. The parameters DIST and DIF to each pair of sensors are calculated to provide the numerical information for the decision-making process of the NCAM-CPN in the subsequent “BehaviorDefinition” and the “DefineMotorCommands” subpages.

Both the “BehaviorDefinition” and “DefineMotorCommands” subpages and their functional hierarchical subpage derivations are essentially two fuzzy logic controller submodels to provide the capability of NCAM-CPN of dealing with uncertaint sensor data. Respectively, the former is responsible for the definition of the most adequate robotic behavior while the latter is in charge of defining the actuator commands in accordance with the robotic behavior instantiated. Both of the information provided by the prior fuzzy controller based subpages defines the output of the NCAM-CPN at this present paper.

The detailed description of the wholly NCAM-CPN structure can be found in Carvalho (2006).

6. NCAM-CPN VERIFICATION BY MEANS OF SIMULATION

In order to verify NCAM-CPN by means of simulation, input data defined by the text file “SensorData” were gradually incremented, knowing in advance the robotic behaviors and the final actuator commands that should have been activated. Thus, the NCAM-CPN could be checked about the occurrence of the three behaviors (follow-path, go-straight and avoid-obstacle), and of the several possible final robot movements.

The text file “SensorData” structure is determined by a sequence of sensor readings grouped in a 9-uple of integer values. The first element of the 9-uple represents the order of the reading, while the other elements correspond to the sensor data rounded to integer values.

```
1`(1,85,84,20,22,19,20,20,21)++ % First reading
1`(2,84,83,21,23,20,19,21,22)++  % Second reading
1`(3,85,83,22,24,20,18,20,21)++  % Third reading
1`(4,84,82,23,25,21,19,21,23)++
1`(5,85,83,24,26,22,20,22,24)
```

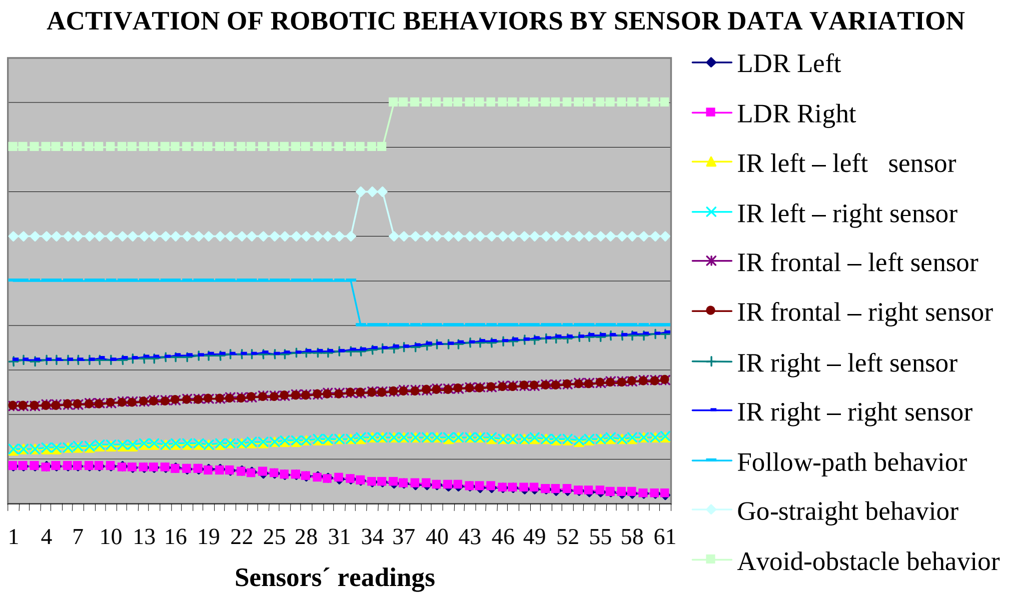


Figure 4. Graph of the activation of robotic behaviors for sensors´ readings during the simulation of NCAM-CPN based on the IR obstacle detection sensor and the LDR path detection sensor data.

Figures 4 and 5 depict two of the resultant graphs that can be obtained by means of NCAM-CPN simulation. The activation of the expected robotic behaviors is plotted in Figure 4. In turn, the final robot movements, as a complement and simultaneity of the Figure 4 graph, are plotted in Figure 5.

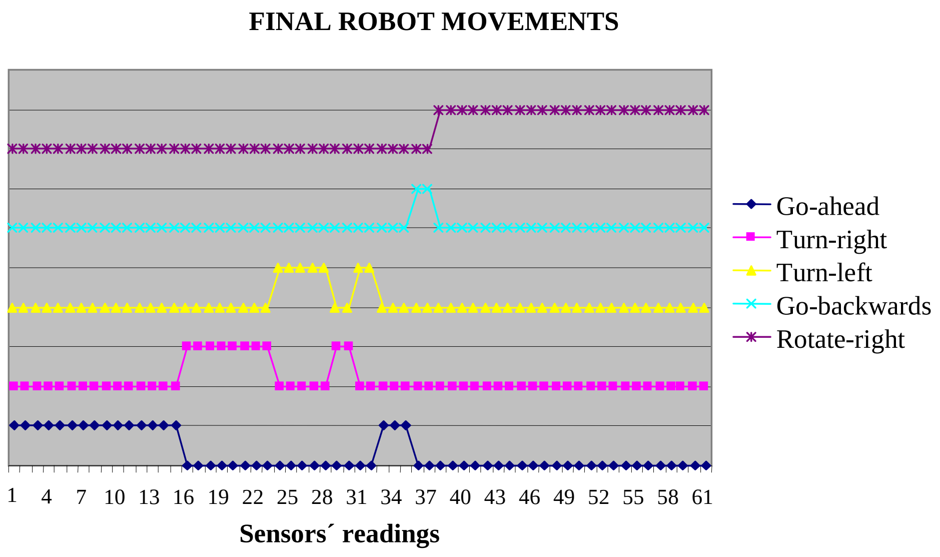


Figure 5. Graph of the resultant or final robot movements for sensors´ readings during the simulation of NCAM-CPN.

In Figure 4, each level represents a behavior state, that is, activated (highest level) or deactivated (lowest level). The input data to NCAM-CPN of each pair of sensor are also plotted, showing the gradual value variation for the verification purposes. In Figure 5, the final movements of the robot are depicted as an output of a function (method) defined in NCAM-CPN, depending on the resultant values of the actuators commands (left and right motors). By the Figure 4 and 5 results, NCAM-CPN is verified, since the prior intended output corresponds to the actual one.

7. CONCLUSIONS AND FURTHER WORKS

Classically, the control of robotic systems has been based on the theory of modern control, particularly concerned with the parameterization of dynamics and kinematics proprieties. On the other hand, artificial intelligence has also been applied in the development of alternative robotic control techniques by means of neural networks, genetic algorithms and fuzzy logic, among others. The classical approach is intimately related to the search for a control law to be verified mathematically in terms of its stability. The AI approach is independent of prior information about the robotic system kinematics and dynamics. The latter is based on machine learning techniques, resulting in robotic control architectures which attempt to be adaptable to external contingencies.

As discussed in Section II and III, the robotic control architectures have been considered as an efficient approach to deal with the control of robotic systems. Nevertheless, the modeling and the simulation of such a complex system demand a computational tool which possesses a solid formalism for mathematical analysis. Owing to this, this work focused on an investigation about the use of high level Petri nets in order to overcome the limitations and the contradictions described by the preliminary work of Carvalho *et al.* (2005).

The effective modeling of the decision-making process as described in Section V and verified in Section VI constitutes a fundamental differentiation from the prior related (or not) works for which its synthesis is the navigational control architecture model or NCAM-CPN. By the latter, the qualitative and quantitative aspects present in an intelligent system based control system were finally formalized. NCAM-CPN is capable of modeling and simulating the sensor data acquisition, interpret them by fuzzy logic controller submodels and defining or changing reactively the actuators commands and robotic behaviors without the necessity of a new Petri net extension definition.

During the implementation of NCAM-CPN, it was possible to verify that its complexity derives from the demanding of the modeling of different and simultaneous levels of abstraction. For instance, a robot which is externally interacting with its environment, collecting sensor data, receiving robotic tasks and answering to those stimuli and, simultaneously, internally defined by its control system, interacting with itself in order to provide the most appropriate and efficient decision-making. This problematic issue was already discussed by Cherkasova *et al.* (1993). In this sense and in consonance with the latter work, it was observed that the use of colored Petri nets and CPN Tools in real robotic control can be limited by computation resources, due to the time and the complexity demanded by the addition of new specifications, new modeling elements to improve NCAM-CPN capabilities. For further work, the use of the object-oriented Petri nets´ formalism defined by Lakos (1997) may pave the way to overcome the drawbacks herein found.

8. ACKNOWLEDGEMENTS

The authors would like to acknowledge CAPES for financial support.

9. REFERENCES

Arkin, R., 2000, Behavior-based Robotics, The MIT Press, Cambridge, USA.

Brooks, R. A., 1986, “A robust layered control system for a mobile robot”, IEEE Journal of Robotics and Automation, Vol. 2, No. 1, pp. 14-23.

Caloini, A. et al., 1998, “A Technique for Designing Robotic Control Systems based on Petri Nets”, IEEE Transactions on Control Systems Technology, Vol. 6, No. 1, pp. 72-86.

Carvalho, H. J. R., 2006, The use of Petri nets for modeling and simulation of a behavior-based robotic control architecture for mobile robots, Masters thesis, School of Engineering of São Carlos, University of São Paulo, São Carlos, São Paulo, Brazil.

Carvalho, H. J. R. et al, 2005, “Modeling of a Navigation Control Architecture for a Mini-Robot using Petri Nets”, In: 18th International Congress on Mechanical Engineering, Ouro Preto, Proceedings of the COBEM 2005, ABCM, Brasília, Brazil.

Cherkasova, L.; Kotov, V.; Rokicki, T., 1993, “On net modeling of industrial size concurrent systems”, Lecture Notes In Computer Science**:** Proceedings of the 14th International Conference on Application and Theory of Petri Nets, Vol. 691, pp. 552-561, Springer, New York, USA.

CPNTools. Available via <http://wiki.daimi.au.dk/cpntools/cpntools.wiki>. [accessed in july, 2006]

Desrochers A., Al-jaar R. 1995, Applications of Petri Nets in Manufacturing Systems: modeling, control, and performance analysis, IEEE Press, New York, USA.

Jensen, K, 1997, Coloured Petri Nets: Basic Concepts, Springer, New York, USA.

K-Team. Available via <http://www.k-team.com/>. [accessed in may, 2006].

Lakos, C., 1995, “From colored Petri nets to object Petri nets”, Lecture Notes In Computer Science**:** Proceedings of the 16th International Conference on Application and Theory of Petri Nets, Vol. 935, Springer, New York, USA.

Lima, P. and Saridis, G., 1996, “Learning Optimal Robotic Tasks”, IEEE Expert, Vol. 11, No. 2, pp. 38-45.

Lima, P. et al., 1998, “Petri Nets for Modeling and Coordination of Robotic Tasks”, Proceedings of the IEEE: International Conference on Systems, Man and Cybernetics, Vol. 1, pp. 190-195.

Milutinovic, D. and Lima, P., 2002, “Petri Net Models of Robotic Tasks”, Proceedings of the 2002 IEEE: International Conference on Robotics and Automation, Vol. 4, Washington, USA, pp. 4059-4064.

Montano, L. et al., 2000, “Using the Time Petri Net Formalism for Specification, Validation and Code Generation in Robot-control Applications”, The International Journal of Robotics Research, Vol. 19, No. 1, pp. 59-76.

Murata, T., 1989, “Petri Nets: Properties, Analysis and Applications”, Proceedings of the IEEE, Vol. 77, No. 4, pp. 541-580.

Murphy, R., 2000, Introduction to AI Robotics, The MIT Press, Cambridge, USA, 466 p.

Porto, A. J. V. et al., 2003, “Robô Agrícola Autônomo (RAM): uma revisão das pesquisas recentes sobre sistemas de navegação autônoma de robôs e veículos agrícolas” (CD ROM), In: Congresso Brasileiro da Sociedade Brasileira de Informática Aplicada à Agropecuária e Agroindústria, 4., Resumo, Editores: Marcos Aurélio Lopes, André Luiz Zambalde, Anais, Porto Seguro, Brasil.

Sousa R. V. et al., 2005, “Compostion, coordination and simulation of reactive fuzzy behaviors for a mobile agricultural robot”, In: 18th International Congress on Mechanical Engineering, Ouro Preto. Proceedings of the COBEM 2005, ABCM, Brasília, Brazil.

Wang, F. et al., 1991, “A Petri-Net Coordination Model for an Intelligent Mobile Robot”, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 4, pp. 777-789.

Wang, F. and Saridis, G., 1993, “Task Translation and Integration Specification in Intelligent Machines”, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 3, pp. 257-271.

10. RESPONSIBILITY NOTICE