MODELING OF A CONTROL ARCHITECTURE FOR A MINI-ROBOT NAVIGATION USING PETRI NETS

Hilano José Rocha de Carvalho
hilanorc@sc.usp.br

Rafael Vieira de Sousa
rafael@cnpdia.embrapa.br

Andrea Ribari Yoshizawa
arys@sc.usp.br

Arthur José Vieira Porto
Escola de Engenharia de São Carlos – USP
Engenharia Mecânica / LAB – Simulação
Av. Trabalhador São-Carlense, 400, CEP. 13566-590, São Carlos - SP
ajvporto@sc.usp.br

Ricardo Yassushi Inamasu
EMBRAPA, Centro Nacional de Pesquisa e Desenvolvimento de Instrumentação Agropecuária
R 15 de Novembro, 1452, CEP. 13560-970, São Carlos - SP
ricardo@cnpdia.embrapa.br

Abstract. *The research in autonomous systems has been influencing the improvement in navigation of mobile vehicles and robots using artificial intelligence techniques. Agriculture and industry are instances of economical segments that may be benefit from the application of those scientific efforts. However, those systems demand control architectures that, most of the time, have a high degree of complexity when having tested by physical implementations. On the other hand, recent works confirm the eficacy of Petri nets for modeling real systems with concurrent activities as well as in planning and controlling of mobile robot tasks. The aim of this paper is to apply Petri nets for modeling and analysing concurrent behaviors present in a simple robotic navigation architecture, which was successfully tested in a mini-robot (Khepera), to apply as reference for implementation in a Autonomous Agricultural Vehicle (VAA). Available software for academic purposes were used for models development and proprieties analysis. The results point that using the formalism proposed for modeling, one is capable of determining control policies, analysing and identifying conflicts in concorrent robotic behaviors with ease in comparison to real implementations.*

Keywords*: Mobile vehicles and robots, Petri Nets, Modeling, Robotic architectures.*

1. Introduction

Nowadays, implementations in robotics are commonplace in the world. Mainly, in the industrial segment, robots have been used successfully as a powerful tool in order to reduce costs and improve efficiency through the last decades. However, there are other economical segments, such as agriculture, about which the research in robotics has been trying to focus. The development of platforms for mobile robots, which may be capable of operating in odd environments - subjected to different natural adversities - aims at providing new approaches for agricultural purposes. That is the case of the Autonomous Agricultural Vehicle project for which this work is related (Porto *et al*., 2003).

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. 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). 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.

A behavior-based architecture for navigational purpose using fuzzy logic was developed and successfully tested in a mini-robot platform (*Khepera*). However, that robotic implementation demands a way of representing the interaction among the data, which influence concurrently in mobile robot decisions. Being capable of knowing in advance the behavior of the mobile robot may facilitates the improvement of that and other architectures to come, especially, about the tasks to be carried out. A way of foreseeing the robotic architectures´ inner interactions, in turn, demands a

modeling tool, which may be capable of representing concurrent activities. Petri nets constitute a powerful formalism for that purpose.

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), and in the development of autonomous robots for unstructured environments, such as mobile robots for agricultural purposes.

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 tranducers (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.

The approach by Montano *et al*. (2000) for the problem of control policy of tasks presents a preliminary distinction of three types of transitions: transition-CODE, transition-TIME and transition-SYCO. The transition-CODE has to do with programming codes associated with activities, the transition-TIME models the occurrence of periodical events or time-outs and the transition-SYCO is in charge of synchronization and task control. These concepts are applied to modeling navigational tasks, especially those related to detection and obstacle-avoidance carried out by sensors (lasers) in path planning. From that pattern, processes associated with sensors are also defined – control and supervision processes – integrating them through a centralized and de-centralized approach. The authors conclude that the de-centralized approach is more efficient for the concerning objective.

Considering the papers presented in this section, this work incorporates the principles of the hierarchical organization as the one proposed by Wang *et al*. (1991) in order to provide a modularized approach. However, the association of the notion of time with the interaction among the subdivisions obtained diverges from Montano *et al*. (2000), since Generalized Stochastic Petri Nets (GSPNs) are applied instead of Time Petri Nets (TPNs). In addition to the use of GSPN theory, the main difference of this work from the ones discussed above is related to the application of Petri net formalism to a behavior-based architecture modeling, considering all of its concurrent aspects in order to make it practical for a real robotic task problem.

This paper is subdivided in five subsequent sections. In section II, the principal aspects of the navigational architecture for a mini-robot platform and its results are discussed. Section III presents the formal definitions of a Generalized Stochastic Petri Net (GSPN) and the software, GreatSPN (GreatSPN, 2005), used for the construction and validation of the Navigational Control Architecture Model (NCAM). The NCAM is described in section IV and applied to a simple task problem in section V.

2. 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 widely applied to behavior-based robots giving them the capacity to handle unusual situations in decision-making process.

A navigational behavior for a mini-robot platform was delevoped using fuzzy logic. All the field tests were carried out in a mini-robot platform, the commercial robot *Khepera*. 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 mo dule 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.

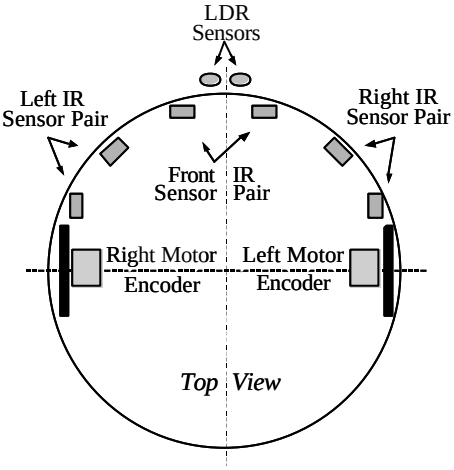


Figure 1. A top view of Khepera-based platform.

3. Petri Nets

3. 1. Formal Definitions

Different extensions to Petri nets can be found in recent bibliography. Depending on the system to be modeled, the basic elements of Petri net assume different functions. That is the case of the interpreted Petri nets. Due to the approach of this work, the type of interpreted Petri net used may associate a specific distribution time delay, commonly an exponential distribution, with the transitions´ firing rule: the stochastic Petri nets (SPNs). Considering Murata (1989), suppose the delay d, associated with transition t, is a non-negative continuous random variable X with the exponential distribution function defined by Eq. (1).

$$F_x(x) = Pr[X \leq x] = 1 - e^{-l_i x} \tag{1}$$

Then, the average delay is given by Eq. (2), where λ is the firing rate of a transition t.

$$\overline{d}_i = \int_0^\infty [1 - F_x(x)]dx = \int_0^\infty e^{-l_i x} dx = \frac{1}{l_i} \tag{2}$$

Based on the principles of a SPN, Ajmone Marsan *et al*. (1995) defines a Generalized Stochastic Petri Net (GSPN) model as a 10-tuple M_{GSPN} = (*P, T, P , I, O, H, W, PAR, PRED, MP*), where M_π = (*P, T, P , I, O, H, PAR, PRED, MP*) is a 9-tuple, the underlying Petri net model or a Petri net model with priority. $W : T \rightarrow IR$ is a function defined on the set of transitions. $P : T \rightarrow IN$ is the priority function that maps transitions onto natural numbers representing their priority level. The priority level, actually, defines another important type of transition in a GSPN model: the immediate transition, which has also the property of firing as soon as it is enabled. Finally, *M* = (*P, T, I, O, H, PAR, PRED, MP*) defines a Petri net model, where:

- *P* is the set of places;
- *T* is the set of transitions;
- *I, O, H* : $T \rightarrow Bag(P)$, are the input, output, and inhibition functions, respectively, where *Bag(P)* is the multiset on P;
- *PAR* is a set of parameters;
- *PRED* is a set of predicates restricting parameter ranges;
- $MP : P \rightarrow IN \cup PAR$ is the function that associates with each place either a natural or a parameter ranging on the set of natural numbers.

The research of Ajmone Marsan *et al*. (1995) culminated in the development of the software called GreatSPN (GreatSPN, 2005), which is capable of modeling and validating a GSPN. Considering its functional aspects, GreatSPN was used for modeling and analyzing properties of the Navigational Control Architecture Model (NCAM) proposed in this work. Simulation results and performance reports are not the focus of this present work.

4. Modeling of a Mini-Robot Navigational Control Architecture using GSPNs

4.1. Modularization of the robotic mobile system

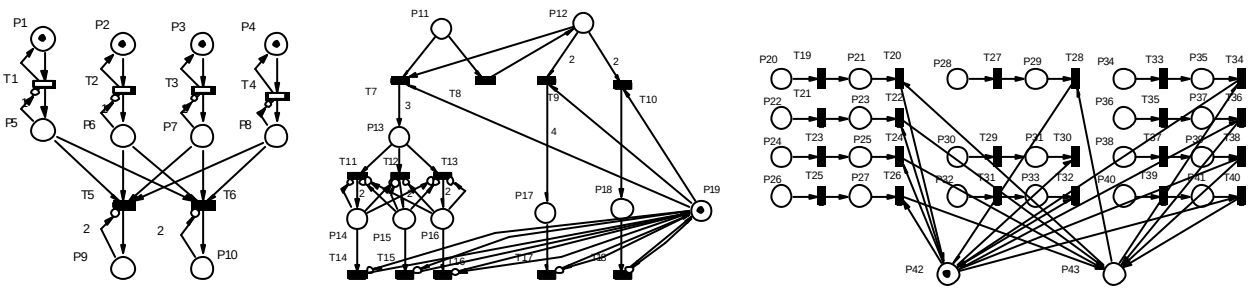
The first step of the modeling consists of the subdivision of the system into three main modules: the sensor module, the fuzzy logic behavior module and the motor command module. This procedure intends to specialize each module in a determined role, facilitating the understanding of the interaction among the different processes that might be present in the robotic mobile system.

Firstly, the firing of transitions t_1, t_2, t_3 and t_4 of sensor module in Fig. 2 models the updating of data (qualitative data – presence or absence) from the four groups of sensors (IR or LDR). This data updating is defined by an exponential distribution function that leads to the definition of exponential transitions with their firings rates λ. For instance, nodes formed by p_1-t_1-p_5 model the supply of data about obstacle and path distances and positions from the IR sensor located on the left of the mini-robot. Besides, nodes p_2-t_2-p_6, p_3-t_3-p_7 and p_4-t_4-p_8 model the IR sensor located on the front, IR sensor located on the right and the LDR sensor, respectively.

Secondly, the fuzzy logic behavior module of Fig. 2 models the decision-making process of the fuzzy behavior arbitration that has to take into account the hierarchy of priorities among the different behaviors. This idea is implemented by places p_{11} and p_{12} that receive tokens from the sensor module, modeling data information checked and supplied about obstacle and path distances and positions. Depending on these important data, the firing of transitions t_7, t_8, t_9 and t_{10} defines which behavior is initiated. Since those transitions are immediate transitions, different values of *P* are defined, representing the association of the concept of priority level.

Finally, the motor command module of Fig. 2 models the motion state of the mobile robot according to the fuzzy logic behavior decision (places p_{23}, p_{25}, p_{27}, p_{29}, p_{31}, p_{33}, p_{37}, p_{39} and p_{41}) and the behavior itself (p_{21}, p_{29} and p_{35}).

Sensor Module *Fuzzy Logic Behavior Module* *Motor Command Module*



SUBTITLES

Places

Sensor Module

P1: IR_L_Data
P2: IR_F_Data
P3: IR_R_Data
P4: IR_LDR_Data
P5: Check_IR_L
P6: Check_IR_F
P7: Check_IR_R
P8: Check_LDR
P9: Check_DIF
P10: Check_DIST

Fuzzy logic Behavior Module

P11: Check_For_Obst
P12: Check_For_Path
P13: Avoid_Obst
P14: Avoid_Obst_L
P15: Avoid_Obst_F
P16: Avoid_Obst_R
P17: Follow_Path
P18: Straight_In_Line
P19: Check_IF_Behavior

Motor Command Module

Behaviors

P21: Avoiding Obstacle
P29: Follow Path
P35: Going Straight

Motor States

P23: FLFR - Going Forward
P25: FLSR - Turning Right
P27: FLBR - Rounding Right
P31: SLSR - Stop
P33: SLFR - Turning Left
P37: BLFR - Rounding Left
P39: SLBR - Turning Backward Left
P41: BLBR - Going Forward

Transitions

Sensor Module

T1 (Exponential): Update_IR_L_Data
T2 (Exponential): Update_IR_F_Data
T3 (Exponential): Update_IR_R_Data
T4 (Exponential): Update_LDR_Data
T5 (Immediate): Get_DIF(IR)_DIF(LDR)
T6 (Immediate): Get_DIST(IR)_DIST(LDR)

Fuzzy Logic Behavior Module

T7 (Immediate): Obstacle_Found
T8 (Immediate): Obstacle_Not_Found
T9 (Immediate): Path_Found
T10 (Immediate): Path_Not_Found
T11 (Immmediate): Obstacle_Found_Left
T12 (Immediate): Obstacle_Found_Front
T13 (Immediate): Obstacle_Found_Right
T14 à T18: Immediate Transitions

Motor Command Module

T19 à T40: Immediate Transitions

Figure 2. Sensor, fuzzy logic behavior and motor command modules: figures and subtitles.

4.2. Integration of the main modules into a Navigational Control Architecture Model (NCAM)

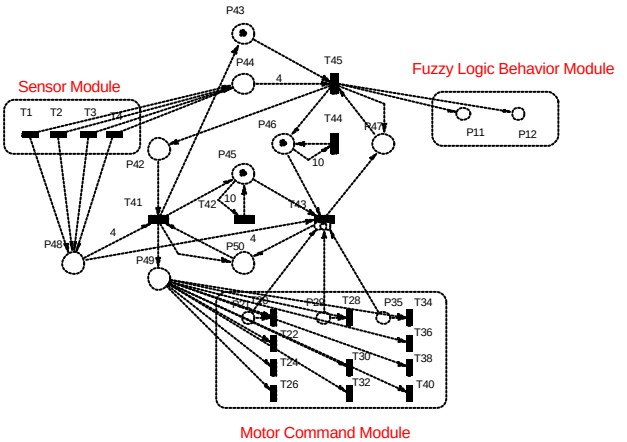
All the three main modules have no functional meaning when analyzed independently. The integration of them constitutes the final conception of a Navigational Control Architecture Model (NCAM) and justifies its subdivision in specialized modules.

First of all, the main control module is defined. It is in charge of associating the three main modules with respect to the rate of sensor data updating (sensor module), resulting in the most appropriate behavior (fuzzy logic behavior module) and the correspondent motor commands (motor command module). Figure 3 shows the interaction of all the main modules according to the main control module, describing a control policy.

Another integrating module proposed is the fuzzy controller module. Based on the input data provided by the sensor module, the fuzzy controller module applies fuzzy logic mechanisms to convert them in another type of data, which will be used by the motor command module. Then, the motions of the mobile robot can be defined to certain conditions that are detected and interpreted.

In Figure 4, the data interpretation is modeled qualitatively, since no data quantitative value is taken into account. This qualitative approach is based on the combinations of the output arcs from transitions t_a to t_c of DIST (FAR, MED, CLOSE) and from transitions t_d to t_f of DIF (NEG, ZERO, POS) values for the obstacle avoidance and path behaviors. As a matter of fact, the uppermost model in Fig. 4 represents a summarized form of four models with their common connection with the other modules. One model for avoid left obstacle, two others for avoid front obstacle and for avoid right obstacle behaviors respectively, and the last one for follow path behavior. The composition of all the four models showed in Fig. 4 constitutes the fuzzy controller module. The indexes range of transitions and places may vary, as described by the subtitles of Fig. 4, due to the fact that different behaviors are represented.

In Figure 5, places from p_a to p_i and transitions from t_g to t_p, derived from the fuzzy controller module of Fig. 4, model the resulting motor motion command (motor command module). Finally, in Fig. 6, the remaining elements and last connections conclude the Navigation Control Architecture Model (NCAM) composition. It is also represented the final layout of NCAM.



SUBTITLES

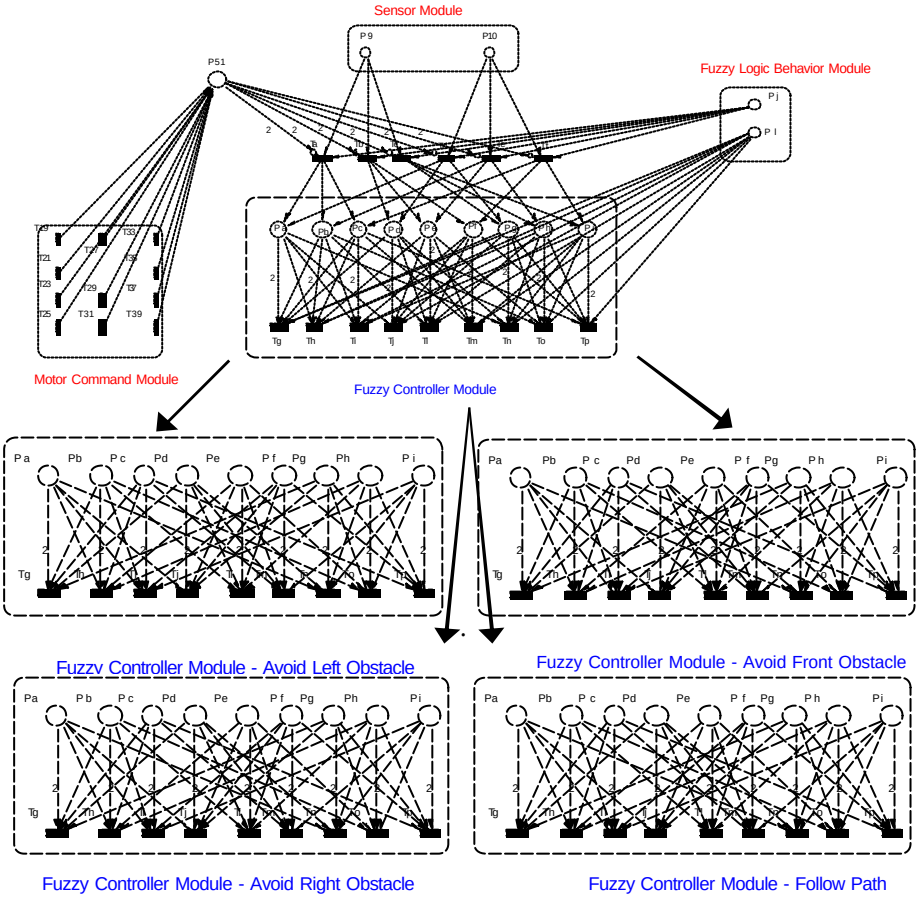
Places

P11: Check_For_Obst
P12: Check_For_Path
P21: Avoiding Obstacle
P29: Following Path
P35: Going Straight

Transitions

T1 (Exponential): UPd_IR_L
T2 (Exponential): UPd_IR_F
T3 (Exponential): UPd_IR_R
T4 (Exponential): UPd_LDR
T20, T24, T26, T28, T30, T32, T34,
T36, T38, T40 e T41 à T45 (Immediate)

Figure 3. Main control module.



SUBTITLES

Places

Pa: Neg-Far
Pb: Neg-Med
Pc: Neg-Close
Pd: Zero-Far
Pe: Zero-Med
Pf: Zero-Close
Pg: Pos-Far
Ph: Pos-Med
Pi: Pos-Close

P9: Check_DIF
P10: Check_DIST
Pj, j = 13: Avoid_Obst
Pj, j = 17: Follow_Path
Pl, l = 14: Avoid_Obst_L
Pl, l = 15: Avoid_Obst_F
Pl, l = 16: Avoid_Obst_R

Transitions (Immediate)

Ta, a = 46: Neg_Avoid
Ta, a = 52: Neg_Follow
Tb, b = 47: Zero_Avoid
Tb, b = 53: Zero_Follow
Tc, c = 48: Pos_Avoid
Tc, c = 54: Pos_Follow

Td, d = 49: Far_Avoid
Td, d = 55: Far_Follow
Te, e = 50: Med_Avoid
Te, e = 56: Med_Follow
Tf, f = 51: Close_Avoid
Tf, f = 57: Close_Follow

Figure 4. Fuzzy controller module.

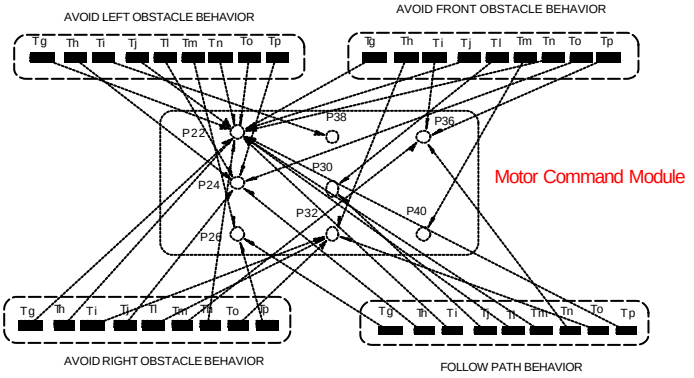


Figure 5. Fuzzy controller module and its connections to the motor command module.

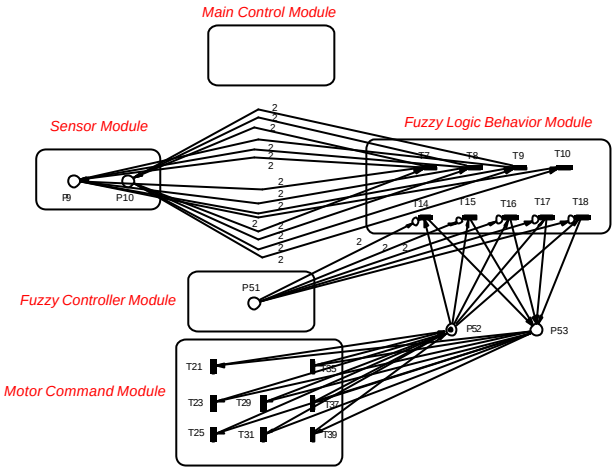


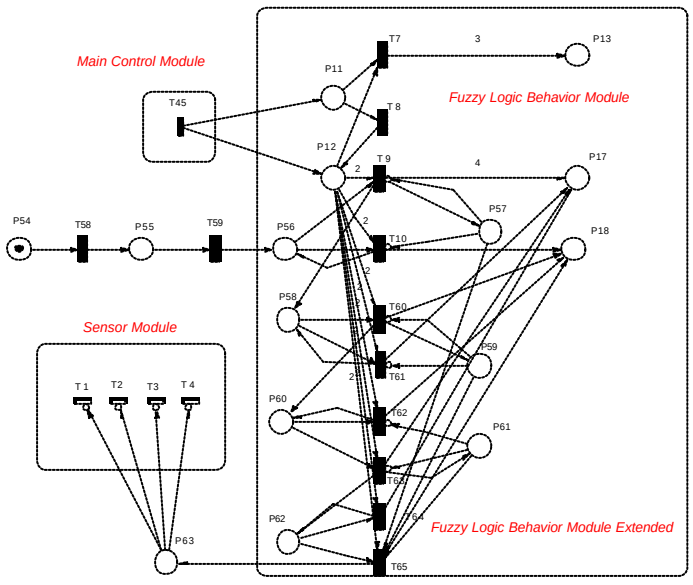
Figure 6. The Navigation Control Architecture Model (NCAM): the final integration among the different modules.

5. Application of the Navigation Control Architecture Model to a simple robotic task problem

The Navigation Control Architecture Model (NCAM) constitutes a *live*, *bounded* and *reversible* GSPN model, as validated by GreatSPN.

A robotic task can be subdivided in a sequence of sub-tasks and these, in turn, will take place according to the behaviors dependent on the successive sensor data. Therefore, NCAM can be applied to a simple task problem in which obstacle avoidance and following path leads the robot to change its own motion commands.

A simple robotic task was applied to NCAM. It consists of four sub-tasks. Figure 7 shows a resultant modeling of the robotic task problem with NCAM. Initially, the robot is in standby. This state is modeled by the presence of one token in p_{54}. The firing of transition t_{58} means the start of a task. The sub-task 1 starts when the robot starts going straight, that is, transition t_{59} fires. Since a path is detected (the firing of transition t_9), the sub-task 2 begins, that is, the robot keeps following the path. The end of sub-task 2 and the start of sub-task 3 take place once the path is no longer present and again, the robot goes straight (the firing of transition t_{60}). Once a path is detected (the firing of transition t_{63}), the sub-task 3 ends and, finally, sub-task 4 begins, that is, the robot follows the path. The end of sub-task 4, which means the path is not detected any more (the firing of transition t_{65}), represents the end of the entire task and the mobile robot stops. Concurrently, the obstacle avoidance system is always checking whether an obstacle is present or not. Since, an obstacle is detected (the firing of transition t_7), the avoid obstacle behavior takes place. Otherwise, if no obstacle is found (the firing of transition t_8), the robot behavior may oscillate between the follow path and straight in line behaviors.



SUBTITLES

Places

P54: Standby
P55: Start_Task
P56: Sub_Task_1
P57: End_Sub_Task_1
P58: Sub_Task_2
P59: End_Sub_Task_2
P60: Sub_Task_3
P61: End_Sub_Task_3
P62: Sub_Task_4
P63: End_Sub_Task_4_End_Task_Stop_Robot

Transitions

T7: Obst_Found
T8: Obst_N_Found
T9: Path_Found, End_Sub_Task_1, Start_Sub_Task_2
T10: Path_N_Found_1
T59: Start_Sub_Task_1
T60: Path_N_Found, End_Sub_Task_2, Start_Sub_Task_3
T61: Path_Found_1
T62: Path_N_Found_2
T63: Path_Found, End_Sub_Task_3, Start_Sub_Task_4
T64: Path_Found_2
T65: Path_N_Found, End_Sub_Task_4, End_Task, Stop_Robot

Figure 7. The robotic task model and NCAM integration.

In Figure 7, transitions t_{60} to t_{65} are also connected to the sensor module by places p_9 and p_{10} the same way transitions t_7 to t_{10}. This connection alternates the double direction of the arcs with respect to the behavior that will take place. That is the special case of the straight in line behavior and is clearly depicted in Fig. 6. This approach enunciates the fuzzy logic behavior module extended due to the application of NCAM to a simple robotic task problem.

6. Conclusions

The Navigation Control Architecture Model (NCAM) validates the behavior-based architecture developed for a mini-robot platform. NCAM also represents the effort of a research which deals with the development of a navigation control archicture for an Autonomous Agricultural Vehicle (VAA) (Porto *et al*., 2003).

Considering the issue of the behavioral priority level, NCAM reveals the concurrent procedure among the obstacle-avoidance and any other possible behavior. Since, NCAM is a Petri net model, concurrency can be easily implemented.

Another important feature of NCAM is with respect to its way of integrating the several modules used to separate the modeling. Since those modules are different constituent parts of NCAM, the different interactions between a mobile robot and its environment can be implemented, simulated and also extended.

Another aspect is that considering the potential of modeling and simulation of a GSPN model, once the values for λ and P can vary, NCAM intends to demonstrate its capability to predict, probabilistically, the different behaviors and motions about which a robot will be subjected to complete its tasks.

Even though the use of modularization is well applied to this paper – considering the fact that NCAM is a GSPN model without any other sophisticated approach – the amount of elements, such as places and transitions, is increased considerably. This main disadvantage of using low-level Petri nets has been overcome with the use of high level Petri nets. This approach constitutes the next step of this research.

7. Acknowledgements

The authors would like to acknowledge CAPES for financial support.

7. References

Ajmone Marsan, M. et al., 1995, “Modeling with Generalized Stochastic Petri Nets”, Wiley, Chichester, USA, 301 p.

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.

GreatSPN, 2005, “Great Editor and Analyzer for Timed and Stochastic Petri Nets”, http://www.di.unito.it/~greatspn/index.html

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.

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.