25th ABCM International Congress of Mechanical Engineering
October 20-25, 2019, Uberlândia, MG, Brazil

COB-2019-1275
ADAPTIVE DISCRETE EVENT CONTROL BASED ON INTEGRATION OF PNRD AND iPNRD APPLIED TO BLOCK WORLD DOMAIN

José Jean-Paul Zanlucchi de Souza Tavares
Gabriel de Almeida Souza
Federal University of Uberlandia, Av. Joao Naves de Avila, 2121, Campus Santa Monica, Uberlandia, CEP 38408-100, Brazil
{jean.tavares, gabrielsouzaworking}@ufu.br

Abstract. *One way to integrate physical and informational world operationally is achieved by Petri Nets combined with RFID (Radio Frequency Identification) called PNRD (Petri Nets inside RFID Database), usually applied to passive agents, and iPNRD (inverted PNRD), suitable to active agents. Despite their similarities, their integration is still an open issue. This work purposes to connect them by applying into robotic bench system similarly to the blocks world domain. The procedure is composed by building a PNRD/iPNRD and physical model for each agent, and by integration of their data so that the robot may autonomously order the blocks in order to find the final global state (stored inside each block's RFID tag); comparing logical and physical information, generating a plan, and executing related actions. Some complications arise due to scattered information, inconsistencies between physical and logical information, which requires exception detection due to partial observability, and the movement restriction of one block at a time. The confrontation of PNRD/iPNRD models with physical positioning can verify the initial, intermediate and final condition and also check for exceptions, which can trigger automatically a new identification sweep, resulting in an adaptive discrete event control.*

Keywords: *Petri Nets, PNRD, iPNRD, Adaptive Discrete Event Control, Robotic Sorting*

1. INTRODUCTION

There is a gap between automated planning and operation control and supervision (Fonseca *et al.* 2016). Although planning is easily formalized, acting is not (Ghallab *et al.* 2014). A question arises related to how to integrate control and automated planning.

Wong and Wonham (1996) propose formal algebraic methods for controlling and supervising discrete event systems in a hierarchy architecture. Nishi and Maeno (2010) apply Petri Nets (PN) to optimize routing in a real industrial problem. However, there is a lack of link between information and physical systems. RFID technology is a suitable interface to deal with physical systems whose control depends on information external to the actuator.

Numerous researchers that relate RFID and Petri Net present solutions in several different areas such as quality management of a process, logistic process modelling, healthcare, control design of flexible cells of manufacturing system, monitoring and control of assembly and disassembly systems, material management among others (Tavares *et al.* 2017). These applications have a low-level connection between Petri Net and RFID, and they focused on the creation of the Petri Net marking generation based on the reading of the tags.

The PNRD (Tavares and Saraiva, 2010) provides a formal data structure to tagged objects, which defines and introduces the process activity into a RFID disperse database. In this way, the tag stores its own Petri net (incident matrix and object actual state), and readers have the control vector associated with the reading activity and another conditioning sentence allowing the automatic object Petri net next state calculation, as well as updating its own state vector after the calculation. Since each tag refers to a unique object, the PNRD must be a safe Petri net, and the calculation of the next state must be a unitary vector. Any result other than a unit vector identifies an inconsistency in the process of tag, and it is viewed as an exception.

Another approach called inverted PNRD (iPNRD) is suitable for active agents, and it changes what kind of information each RFID component stores, that is, the tag stores the control vector and additional historical data and readers have the incident matrix and object current state (Fonseca, 2018).

In this paper PNRD and iPNRD models a high-level system (iPNRD) and a lower-level object model (PNRD) to represent and search for an optimal transition firing sequence in order to satisfy each individual passive agent request in a blocks world domain with three blocks. The methods used here are appropriate for small models, as the search problem is NP-hard in a worst case (Gupta and Nau, 1992), but serve as a basis for approaching models that can deal with bigger state spaces. The system also possesses an adaptive nature, as corrections to itself happen in case an exception is detected.

The blocks world (Gupta and Nau, 1992) is one of the most common planning domains in AI research. The goal is to move the blocks with an actuator from the initial state to the desirable state. Only one block may be moved at a time: it may either be placed on the table or placed atop another block. Because of this, any blocks that are, at a given time, under another block cannot be moved. The blocks world's restrictions, actions and possible configurations can be modeled through different paradigms. Finding an optimal route means to change the blocks state to the objective with minimal cost, where cost is generally the total fired transitions. In this work cost is defined in the common sense, with unitary cost for all transitions, the model paradigm being full state representation using PN.

Figure 1 presents three blocks with RFID tag, one robotic arm with RFID reader embedded; and the corresponding Petri Net of this case with 13 places and 30 transitions. Each transition refers to a movement of a block, for instance, *miX2Y* means movement number *i* of block *X* to position *Y* (on table – *T* or another block). Places identifies the positioning of blocks, in example, *AB_C* means that block *A* are on *B* and blocks *B* and *C* are on the table.

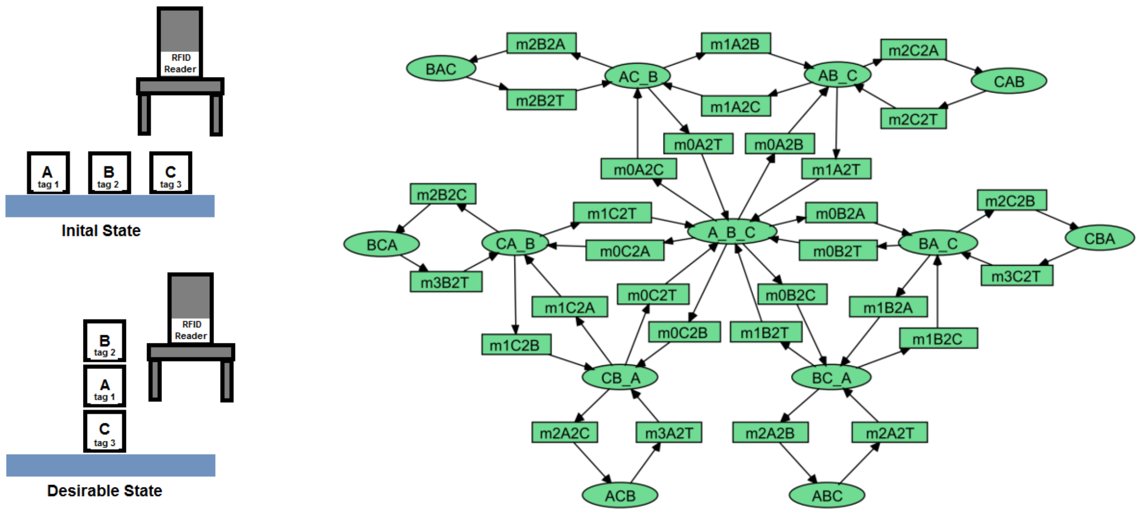


Figure 2. Three Blocks and One Arm with RFID Components; and the system iPNRD model.

2. PNRD AND iPNRD FUNDAMENTS

RFID is a technology based around tags and readers, where each tag emits an unique identification string, as radio signal, and readers can identify and access other info stored inside it. RFID is used because it is a sensory system endowed with individual identification and can store information about the marked item expected attributes and state. It is widely used in industry to track individual products and processes (Tavares and Saiva, 2010). RFID components are tags and readers.

According to Murata (1989), PN consists of a conceptual graphic and mathematical model. PN provides the base for graphical notation and basic primitives for concurrence, communication and synchronization modeling. Formally, an elementary PN is defined as the quadruple $PN=(P,T,F,M_0)$, in which P is a finite set of places, T is a finite set of transitions where $P\cap T= \emptyset$, $F\subseteq(TxP)\cup(PxT)$ is the flow relation, and M_0 is the initial marking.PN can be represented mathematically as a matrix equation presented in Eq. (1), where $[M]_{k+1}$ is the next marking, $[M]_k$ is the current marking, $[A]^t$ is the adjacent matrix and $[u]_k$ is the firing vector.

$$[M]_{k+1} = [M]_k+[A]^t.[u]_k \qquad (1)$$

In this paper PNRD places represent block state, iPNRD places store system state and blocks related positioning; PNRD and iPNRD transitions is related with actuation, system dynamics and motion. There will be only one marking at any time and it exhibits the current state or position for the system (in the iPNRD model) and each block (in the PNRD one).

The PNRD approach (Tavares and Saraiva, 2010) applies PN as a formal data structure distributed across RFID components. In this way, tag PNRD data can be defined by the object incidence matrix $[A]^t$ and its current state $[M]_k$. The RFID antennas and readers are linked to the firing vector $[u]_k$, so readers can perform Eq. (1) during tag data capture. If the result has an "-1" component in $[M]_{k+1}$ it raises automatically an exception. The iPNRD approach (Fonseca, 2018) inverts the RFID components storage, it means, tag stores the firing vector $[u]_k$ and additional data, while readers holds the incidence matrix $[A]^t$ and its current state $[M]_k$. Figure 1 shows PNRD and iPNRD schema.

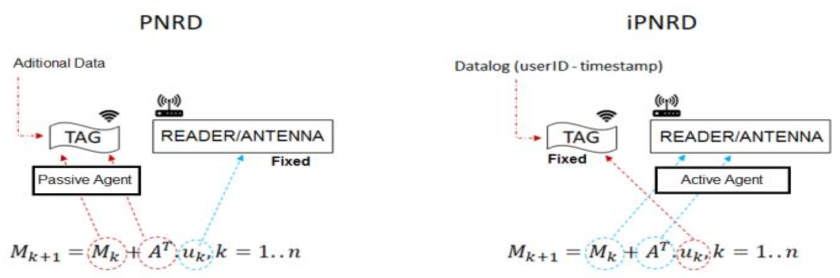
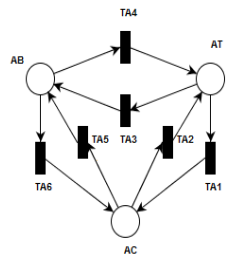

PNRD
Aditional Data
TAG
READER/ANTENNA
Fixed
Passive Agent
$M_{k+1} = M_k + A^T . u_k, k = 1..n$
iPNRD
Datalog (userID - timestamp)
TAG
Fixed
READER/ANTENNA
Active Agent
$M_{k+1} = M_k + A^T . u_k, k = 1..n$

Figure 3. PNRD and iPNRD schema adapted from Fonseca (2018).

3. PNRD AND IPNRD INTEGRATION

In this work the tags are fixed to the blocks, where the PNRD is a net contained by each passive element that has information on the initial and desired state. For this Blocks World specifically the resulting PN is composed by 3 places and 6 transitions for each passive agent as shown in Figure 3. So, for example, block A can be atop block B (place *AB*), C (place *AC*) or the table (place *AT*). For each PNRD transition there are corresponded iPNRD transitions. The relationship between PNRD and iPNRD transitions can be represented directly. For instance, if the robot is executing *M0A2C* it corresponds to T_{A1}, and after robot perform the movement, the RFID reader can trigger PNRD transition on the Block A from *AT* to *AC*. PNRD for block B and C are similar.



PNRD Transition	iPNRD Transition
T_{A1}	M0A2C, M2A2C
T_{A2}	M0A2T, M3A2T
T_{A3}	M0A2B, M2A2B
T_{A4}	M1A2T, M2A2T
T_{A5}	M1A2B
T_{A6}	M1A2C

Figure 3. PNRD regarding block A and iPNRD transitions relationship.

The iPNRD represents the world by the active agent perspective containing information about the global system state as the robotic arm may actuate about any point in the system. This PN does not take path planning into consideration directly. This iPRND is represented in Figure 1. Each transition in the iPNRD contains a hidden, lower level, Petri Net that maps the active agent domain to the physical grid of operation, where motion planning can be performed. Mapping tables to integrate from one layer to the other are required, as a transition in the iPNRD means a change of occupied position in the grid by a block.

For this paper the blocks world has a 3x3 discrete position grid with three different blocks with unique identity. For each block there is a specific column where they may be placed on the bench, the attribution order being column 1 for block *A*, 2 for *B*, and 3 for *C*. The physical grid is discretized as a PN containing an extra line on top for block movement, giving a total of 12 places. The resulting Petri Space is restrained by the column change being enabled only to the top line, to logically avoid collisions. The resulting physical grid is represented in Figure 4, where each position is identified by its column and line.

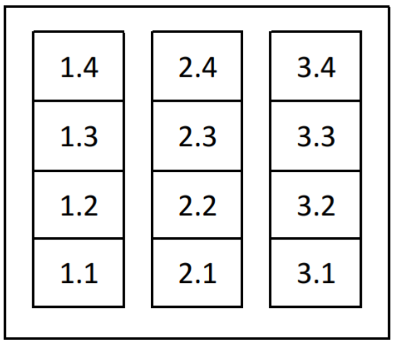


Figure 4. Physical grid with position identity.

It is mandatory to inspect whether the physical arrangement is equivalent to the PN representation regarding the initial state, to rectify for exceptions. This can detect if the blocks information is incoherent or there was external action. This is a sub process in the identification sweep explored in section 4.

The integration is done by union of information in both system levels, by request of the passive agents, the problem being solved taking the logical and physical models, hence each layer restraints the other and is also restrained by the modeling choices. The two main levels are the physical domain, where the robot and the actual blocks rest; and the informational domain, where the models, tables and search algorithms lay, also limited by computer capability. The whole solution is based on retrieving information from one layer to the next so that the Sorting Engine may correctly manipulate the system.

In the physical domain, the active agent is a robotic arm. According to Guérin (2018) there are two distinct trajectory control methods, the first one deals with dynamic treatments for fixed robots, like inverse kinematics for pre-established spatial coordinates, which is very useful with well managed object disposition, but it requires greater environment control in order to work well. In more flexible applications it might be required the use of sensors, automated planners and controllers for adaptive action, like vision systems and machine learning methods, that can be computationally expensive and require training data. This paper uses the first approach.

This work adds a discrete event controller to drive the trajectory planner, integrating the iPNRD transitions to a group of physical movements, it means that the course of action is given by a planner based on iPNRD transitions, being on an abstract level above the robot path control. This creates a hierarchical structure (Wong and Wonham, 1996), where the system discrete event controller determines the objectives for the lower level physical controller. This method is easier to implement and has inferior computational cost when compared to machine learning methods for sorting, being more flexible than a pick-and-place methodology.

As this case study has a simple and small model and the whole state space is represented in a PN, the optimal solution can be found using graph search algorithms. For a more complex model this solution requires different modeling and approach, as an example logical representation and automated planner integration.

4. ADAPTIVE CONTROL OF PNRD/iPNRD BLOCKS WORLD DOMAIN

To plan and execute the sorting job there is need for more processes. To start the planning it is necessary to feed initial and final objective states into the Engine. This information is acquired through identification process and query into a conversion table. Then, after the Sorting Engine executes the plan it is crucial to rectify if the achieved state is equal to the proposed objective state. If no problem is detected the tags are rewritten and a new job begins whether required. This is the process flow in a case where no exception arose. In order to adapt the system, an else statement for each conditional must be designed to deal with possible problems.

The passive objects initial and desired states are obtained through inspection in the physical grid. This identification is performed by sweeping through all columns from the lower position and moving upward, to identify if a block is present in the discrete position and its initial and desired PNRD states. The process' beginning is at position 1.1 (Figure 4) reading until 1.3, repeating for the following columns. When all three columns are scanned, the system will process the PNRD data (tag id, initial state and current state) and the physical position obtained during the reading of each tag. The processor uses position data to infer about initial iPNRD state, and PNRD data to query about the system desired state in the iPNRD, generating the sort objectives if it exists (else it is an exception). It is expected to identify three distinct blocks. It is possible if and only if there is no exception, which occurs if the data identified as initial state of the tag is distinct from the physical data, or when the final state identified in each tag is incompatible, for example if the final state of Block *A* is on Block *B* (PNRD final state *AB*) and the final state of Block *B* is on block *A* (PNRD final state *BA*). This process repeats itself until acceptable states are identified.

Transitions in iPNRD must be converted to a group of transitions of the physical PN, translating the iPNRD transition into sequence of initial and final points. All transitions in the physical PN are converted into a command for the robot driver. This work uses BFS (breadth-first search), that will expand all places following a FIFO (First In First Out) list. This algorithm is optimal for uniform cost arcs (Russel, 2010). The time and space complexity is *O(TP)*, where *T* is the amount of transitions and *P* the total number of places, because the PN is represented as a place-transition incidence matrix and BFS usage. Using a heuristic search might give significantly better results for bigger state spaces, but as the total amount of blocks in this case (3 Blocks) is small, there is no advantage in implementing a more complicated algorithm. The planning and execution phases are further discussed in section 5.

With the dynamic sequence generated, it is possible to start the execution through communication between the involved machines. This interfacing can be done using many different technologies, like serial protocols or online. The system can be adapted to be processed in a distributed or monolithic manner. The command string is interpreted and used by the robot driver to achieve the expected states.

An inspection is fundamental to feedback on the reached state, closing the loop for an adaptive control. This inspection is equal to the identification process and a query is performed to confront the blocks physical disposition with the original desired iPNRD state. If an exception is identified, a new job is started to correct the exception. Else a PNRD rewrite must be performed to update the tags data. This process is performed by moving to each tagged position and transmitting the new current state for each tag. Finally, a job finish conditional is tested to end the sorting or move to the next sequence.

The processes in the adaptive control flow are presented in Figure 5, and they consist of:

- An external command to begin the operation, such as a supervisor boot-start;
- The system performs a sweep to locate the blocks using the arm and mounted RFID reader;
- Identification of iPNRD states through PNRD data and physical position info;
- Perform a planning and execution macro process, the Sorting Engine;
- Another sweep is requested to rectify about any exceptions;
- A PNRD rewrite to each tag is done;
- The system verifies whether the job list is finished. If not, it performs another sequence.

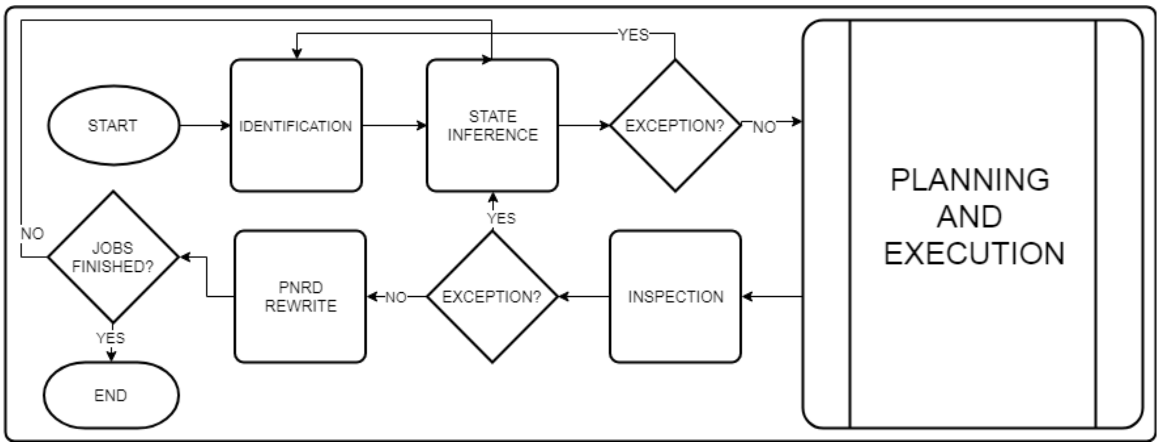


Figure 5.Automatic sorting job flowchart.

5. PLANNING AND EXECUTION MODULE

Nishi and Maeno (2010) apply Petri Nets to optimize routing without taking into consideration the kinematic control of the active agents. This work has a different approach where the physical layer control is directly treated. The iPNRD state space is searched, using BFS, to generate an optimal plan for going from an initial state (set of all PNRD actual state) to a final state (set of all PNRD desirable state); but it is not enough to define the robotic motion system. As there is a hierarchy between the system space state and the physical discrete controller, their relationship can be explicitly modeled in tables, generating sub objectives (intermediate states from the initial to the final state) inside the Petri Space. This flow relation is shown in Figure 6 as the macro plan which defines the Petri Space sub objectives, the Petri Space search or high-level trajectory definition, going to concatenate the required robotic movements, an interpreter for this information to generate the command of robotic drivers.

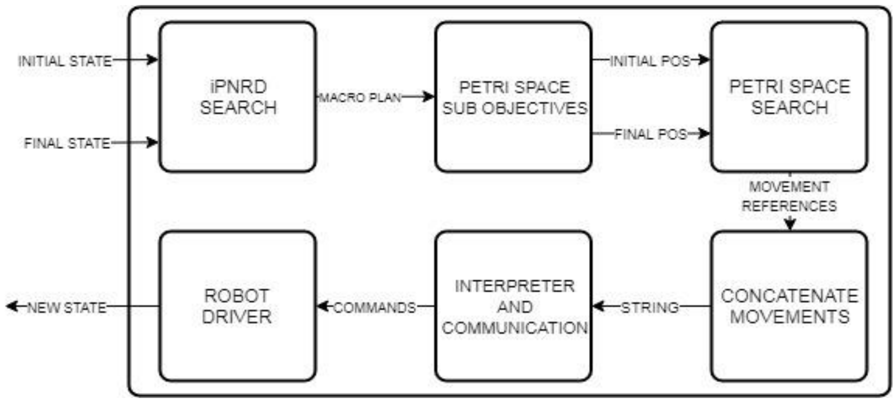


Figure 6.Sorting Engine sub processes.

The BFS iPNRD search obtains the physical domain plan, the macro discrete position controller for the system. This process returns the Petri Space transitions firing sequence, it means, the movement for the Petri Space positioning grid. The recovered transitions need to be correctly concatenated and the robot gripper claw actions must be identified in the correct moment. The concatenation is done by searching from a reference position taking account the required intermediate positioning. The claw is closed whenever it is transporting a block, else it is open. After the concatenated firing sequence is generated it is converted into a readable string for the interpreter in the microcontroller, which is sent through a serial connection to the robot motor drivers.

Each character in the string corresponds to a command for the robot driver. All Petri Space transitions lead to a specific place and all places are abstract representations of a material discrete position. By relating the command to a discrete position the robot driver can correctly alter all servomotors references to reach the desired state, orienting the robot correctly. This process is repeated for each character until the whole command string is processed. When the driver ends a new iPNRD state is reached. Then the after processes described in section 4 may happen to confront for realization of objectives and job, and to rewrite the data in each tag.

6. ADAPTIVE DISCRETE CONTROL EXAMPLE

This example has *ACB* initial state (Figure 7) and *AC_B* as final state (Figure 8). First, the system must be booted by a supervisor, then it will proceed to sweep through the discrete positions identifying the blocks by their unique IDs and get their desired *PNRD* state. The robot will execute this process as described in section 4. The initial identified Petri Space is given by the alpha-ordered triple *PS = {2.3, 2.1, 2.2}*. Querying in a table it is possible to infer that this triple is correspondent to *iPNRD* state *ACB*. The current *PNRD* state can be checked to conclude about changes in the expected state. The data about the blocks desired position is previously added in the *PNRD*, and it is collected during the initial sweep. Thus desired *PNRD* set is *PO = {AC, BT, CT}*, correlating with *AC_B iPNRD* state. The next step is to feed the initial and objective logical states to the sorting engine as {*ACB*, *AC_B*}, in order to determine the course of action in the *iPNRD* state space.

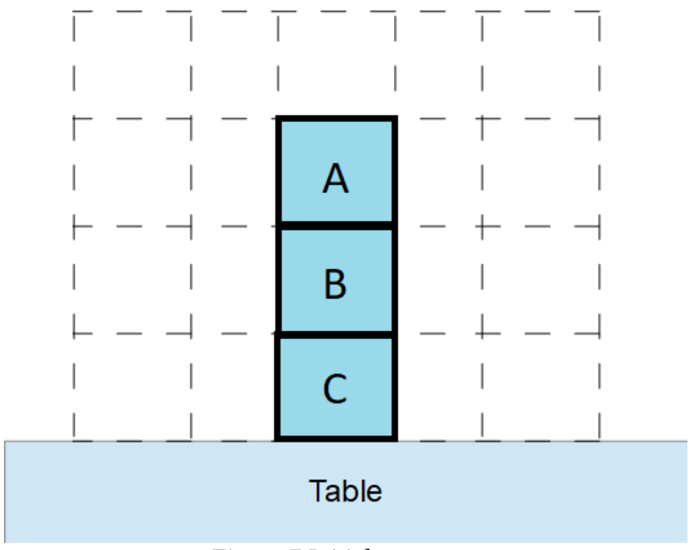


Figure 7.Initial state.

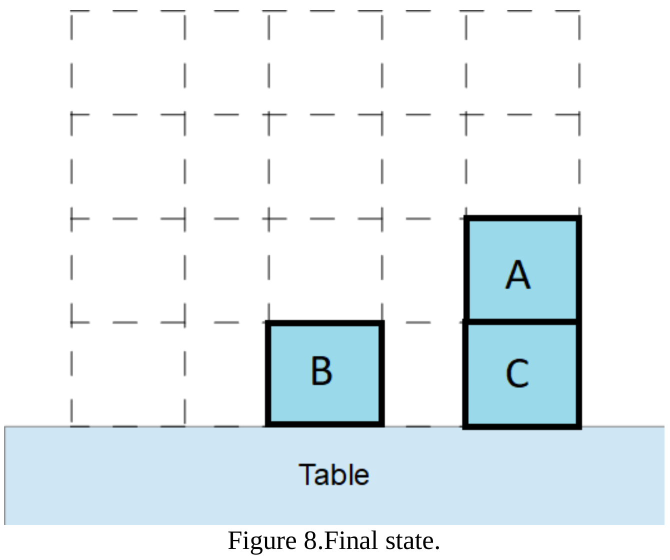


Figure 8.Final state.

The result of a BFS search is the sub objectives list and transition firing sequence. In this case, the best path to reach the objective state is, in order, through *A_CB*, *A_B_C* and finally to *AC_B*. These logical transitions must relate to Petri State transitions in order to actuate on the material world. This is done by performing a query in a relationship table, relating the logical transition to a initial and objective Petri Space places.

To move, the robot must leave a reference initial position, defined as 2.4, to the initial position of the first subobjective with the gripper open. The first subobjective is to change from *iPNRD* state *ACB* to *A_CB*, which translates into moving block *A* from 2.3 to 2.1. To move from 2.4 to 2.3 Petri Space transitions must be searched and fired in sequence. The next step is to grip the block. This action is provided in the database, because logically the block has to be caught and moved. In sequence, transitions moving to 2.4, 1.4 and 1.1 must occur in order. Reaching its final position the arm can open. This ends the movement sequence to reach the first subobjective.

Beginning the second subobjective, it is required to move from the present position to the following initial position. Another search is performed to connect positions 1.1 and 2.2. The transition sequence moves the gripper to places 1.4, 2.4 and 2.2. The process now reiterates until completion, closing the arm and satisfying the second sub objective, then concatenating for the third sub job. When sorting is completed the robot returns to the reference place.

The robot driver just interprets the received string and executes it. A position request is interpreted as follows: A character is received in the Serial channel; This character is related to a specific place; To reach this place servomotor references must be given. Supposing a system with 5 controlled joints, the reference for control can be represented by the five tuple $SR = \{S_1, S_2, S_3, S_4, S_5\}$, where S_x is the reference angle for servomotor X.

After returning to its reference place, a new iPNRD state has been achieved. The inspection protocol must be followed to certify that no exception occurred. In case it did happen, the job is repeated with the new initial iPNRD state and same objective state being fed into the Sorting Engine. Else the PNRD rewrite protocol is performed. When done, the job list is checked to see if all jobs were successfully actuated. If successful the process is over, else the next job is processed.

7. DISCUSSION, CONCLUSIONS AND FURTHER WORKS

This paper presented how to generate an adaptive control based on PNRD and iPNRD integration applied in blocks world with 3 blocks. PNRD is the block information source while iPNRD is the system information one. Both could be checked against the physical position, generating a robust solution. In each operation of the adaptive control it is indispensable to rectify the model. In special on the physical level, the adequate fit for positions and trajectory, with uncertainty and precision levels inside a certain tolerance are essential. To build the models correctly and to realize the system it is fundamental to, in each step, certify that what was built is accurate and done right.

Although this solution has a complete model in iPNRD, for more complex systems (5 blocks, for instance), it could be integrated with automated planners. But when dealing with automatic model generators and bigger systems this control gets harder, and it must be further investigated. The complexity issues are constraints that direct the system modeling and solving in a certain direction. As blocks world complexity grows exponentially with the amount of blocks (Gupta *et* Nau, 1992), using certain tools, algorithms and modeling paradigms can quickly become impractical. There is also the dynamical and geometrical freedom aspect.

Regarding dynamics and concatenation of movement, there is much that can be done to minimize the amount of movements done. The physical PN transition could be expanded enabling extra transitions depending on the certainty that no collisions will occur given a specified trajectory. There are dynamical and geometrical issues regarding the blocks disposition and manipulation. In order to successfully realize the actuation and sensing the system must be precise or robust, meaning that motors torque, potency and control, material strength analysis, sensor signal studies and physical references to assure the fidelity of the driver system motion might be required. In this work we used absolute geometrical references for block and robot placement and attitude.

There can be many different types of exceptions through the process of solving the system, assuming every model and table was created correctly. Exceptions in the system identification might mean that a block is missing or an RFID element is malfunctioning. In the state generator it means that the desired final state does not exist in the model because there is an interest crisis within the PNRD that causes a confrontation. Or the blocks require a physically impossible configuration. The exception identification generates the adaptive event control, meaning that if any failure or external actuation is identified this approach is able to solve it automatically. Exception integration with FMEA (Failure Mode and Effect Analysis) is a future work interest.

8. ACKNOWLEDGEMENTS

This paper is supported by CNPq, CAPES, FAPEMIG and UFU, and it is dedicated to Prof. Dr. José Reinaldo Silva.

9. REFERENCES

Fonseca, J.P.S., 2018. *High-level Petri nets and inverted PNRD associated to mobile robot control: an approach to search and rescue on trails and crossings*. Ph.D.thesis. Universidade Federal de Uberlândia, Uberlândia, Brasil.

Ghallab, M., Nau, D.S., Traverso, P.: The actors view of automated planning and acting: A position paper. *Artificial Intelligence 208,* 1–17 (2014). https://doi.org/https://doi.org/10.1016/j.artint.2013.11.002, http://www.sciencedirect.com/science/article/pii/S0004370213001173.

Gupta, N., Nau, D.S.: On the complexity of blocks world planning. *Artificial Intelligence 56*(2), 223–254 (1992). https://doi.org/https://doi.org/10.1016/0004-3702(92)90028-V, http://www.sciencedirect.com/science/article/pii/000437029290028V.

Guerin, J., Stephane, T., Nyiri, E., Gibaru, O.: Unsupervised robotic sorting: Towards autonomous decision making robots. International *Journal of Artificial Intelligence and Applications* (IJAIA) 9(2), 81–98 (03 2018). https://doi.org/10.5121/ijaia.2018.9207.

Murata, T., 1989. Petri Nets: "Properties, analysis and applications". *Proceedings of the IEEE,* v. 77, n.4, p. 541-580.

Nishi, T., Maeno, R., 2010. "Petri Net Decomposition Approach to Optimization of Route Planning Problems for AGV Systems". *IEEE Transactions On Automation Science And Engineering*, VOL. 7, NO. 3, pp 523-537

Russell, S.J., Norvig, P., Davis, E., 2010. *Artificial Intelligence: A Modern Approach.* Prentice Hall, New Jersey, 3rd ed.

Tavares, J. J. P. Z. S., Murofushi, R. H., Silva, L. H., Silva, G. R.: Petri Net Inside RFID Database Integrated with RFID Indoor Positioning System for Mobile Robots Position Control. In*: Proceedings of the International Workshop on Petri Nets and Software Engineering* (PNSE'17), pp 157-176, CEUR-WS.org, Zaragoza (2017).

Tavares,J.J.P.Z.S., Saraiva,T.A., 2010. "Elementary Petri net inside RFID distributed database (PNRD)". *International Journal of Production Research* Vol. 48, Issue 9, pp. 2563-2582.

Wong, K. C., Wonham, W. M., 1996. "Hierarchical Control of Discrete-Event Systems". Discrete Event Dynamic Systems, NO. 6, pp 241-273.

10. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.