Simulation and Analysis of Route Plans for Autonomous Guided Vehicle Systems

Rozália Lakner
Alba Regia Technical Faculty
Óbuda University
Székesfehérvár, Hungary
lakner.rozalia@amk.uni-obuda.hu

***Abstract* – Automated Guided Vehicle Systems (AGVS) are designed to transport objects automatically, efficiently, and safely between locations. In AGVS, vehicles follow a defined route, which is determined taking into account, for example, collisions and deadlocks. To do so, an important part of the operation of AGVS is the examination of the planned routes in order to detect potential problems. The paper demonstrates the application of Colored Petri nets for simulation and analysis of AGVS and for detecting and avoiding deadlocks and collisions.**

Keywords – Automated Guided Vehicles System (AGVS), Colored Petri net, simulation and analysis, deadlock.

I. INTRODUCTION

Autonomous Guided Vehicles (AGVs) are used to automate the transport of parts and materials within a production facility, either indoors or outdoors. The study of an AGV System (AGVS) involves several activities such as layout planning, synthesis of transport activities and feasible pathways, determining the number of AGVs, deadlock prevention and scheduling transport activities, so the design and further performance analysis of AGVS can be quite complex. In general, discrete event simulation is used as the primary tool for modelling and analyzing AGV systems [1, 2].

Petri nets [3] are a modelling formalism that allows a natural representation of discrete event systems. A Petri net has both graphical and mathematical formalism for describing concurrent, asynchronous, distributed systems [4]. Over the past half century, various extensions to Petri nets have been developed, e.g., colored, timed, hierarchical, and many successful applications have been made in a wide variety of domains. These methods support both the modeling and the analysis of systems in various application areas, including process engineering, manufacturing systems and computer science.

The current work presents the construction and application of Colored Petri net models for simulation and formal analysis of AGV systems.

II. COLORED PETRI NETS

Colored Petri net (CPN) is a Petri net extension that allows tokens to be distinguished by assigning a data value to the tokens [5]. Places in a CPN can be assigned types, and tokens have values taken from the types associated with the places. These values, which may represent complex types, are called colors. The colors assigned to places can be specified using the place color set.

A. Formal definition of Colored Petri nets

Colored Petri net formally can be described by an octuple in the form $CPN =< P, T, A, \Sigma, C, E, G, I>$, where P is the finite set of the places, and T is the finite set of the transitions, with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$. A defines the arcs from places to transitions and contrariwise such as $A \subseteq (P \times T) \cup (T \times P)$. A Petri net is also known as a place/transition net, where places play a role as passive elements, e.g., states, conditions, and transitions correspond to active elements, e.g., events, actions, movements. Σ defines a finite set, called color set, and C is a color function. The color set is defined by specifying a data type, which can also be a multiset. The color function specifies the type of tokens it can accept for each place, i.e., it assigns to the place the color set of possible tokens there, C: $P \rightarrow \Sigma$. The arc expression function, E, is used to specify the condition for allowing transitions and the consequence of firing transitions, E: $A \rightarrow e$ (where e is an expression). The value of the arc expression function must correspond to the color sets assigned to the corresponding vertices. G defines the guard function, G: $T \rightarrow g$. The guard functions (or guard conditions) allow additional constraints to be specified as eligibility conditions for transitions, which are expressions interpreted on multi-sets. The value of a guard functions is of Boolean type (true, false), and the transition is allowed if the evaluation value is "true". The initial state of the CPN defines the initial marking (distribution of colored tokens), which assigns to each place the multi-set of its associated color set, I: $P \rightarrow i$.

Petri nets can be described as bipartite graphs, where the places are represented by circles and the transitions by rectangles. The places and transitions are connected by weighted directed arcs, denoted by the arc expression functions.

B. Simulation and analysis of Colored Petri nets

In the state M of a Petri net, the transition t is enabled, i.e., firing is possible, if all of its input places contain at least the number of tokens defined by the weight function, i.e., $\forall p \in \bullet t$: $M(p) \geq w(p, t)$. When a transition t fires, a defined number of tokens are consumed from its input places and produced at the output places, so that the state of the net changes, producing a new marking M' as follows: $\forall p \in \bullet t$; $\forall p \in t\bullet$: $M'(p) = M(p) - w(p, t) + w(t, p)$. The simulation of these two basic rules of enabling and firing transitions presents the dynamic behavior of a Petri net. Note that the execution of a Petri net is nondeterministic, i.e., if several transitions are enabled at the same time, any one of them can fire, and different sequences of transitions can lead to different states from the same initial state.

Petri nets are also used for system analysis, where various system properties can be investigated, e.g., reachability, boundedness, reversibility, liveness, deadlock. Most of these properties can be investigated using the reachability graph, which contains all possible states of the Petri net and is constructed from a sequence of all firing transitions from the initial state M_0.

The reachability graph provides information about system properties, such as

- Reachability - The state M is reachable, if there is a sequence of firing transitions from the initial state to M. If M is found as a node of the reachability graph, then M is reachable, otherwise is not.
- L_1 liveness - The Petri net is L_1 live if each transition fires at least once in some firing sequence from the initial state. If all transitions can be found as arc labels in the reachability graph, then the Petri net is L_1 live, otherwise it is dead.
- Liveness - The Petri net is live if all transitions can fire from any reachable states.
- Deadlock - A reachable marking for a Petri net where no transition can fire.

The properties of Petri nets can be marking-dependent, in other words, behavioral, which depend on both the structure and the initial state of the Petri net, and structural, which are independent of the initial marking. Behavioral properties, e.g., reachability, liveness, deadlock, can be investigated in terms of a reachability graph containing the state space of the Petri net constructed from the initial state M_0. The main drawback of these studies is their exhaustive nature, as the reachability graph can be very large, and the time and space requirement increase exponentially with the number of places. This type of analysis is therefore computationally hard.

III. Simulation and Analysis of AGVS with CPN

The chapter presents the CPN model developed for modelling of the route plans for AGVS, as well as the simulation and analysis of the model. The paths of the AGVs were determined as a synthesis task using a P-graph framework [6], and the possible paths were analyzed using CPN. The Petri net model of the AGVS implemented by CPN Tools [7].

A. The CPN model of the AGVS

To determine the AGV routes, first, it is necessary to know the environment. Fig. 1. illustrates the AGV layout under study, where three rooms (*A*, *B* and *C*) are defined, with gateways (*G1*, *G2* and *G3*) and with three special stations (*S* charging station in room *A*, *T1* and *T2* pick-up and delivery stations in room *B* and *C*). AGVs can travel between different points to pick up and deliver objects along predefined routes. For the safe movement of AGVs, safety zones must be established, which must be defined considering the size of the vehicle. These zones are shown in green on the left side of Fig. 1, together with their specific locations. This environment, implemented using the CPN-Tools CPN simulator, is shown on the right side of Fig. 1., where the specific locations and the possible movements of AGVs between two neighboring locations correspond to the places and transitions in Petri net, respectively.

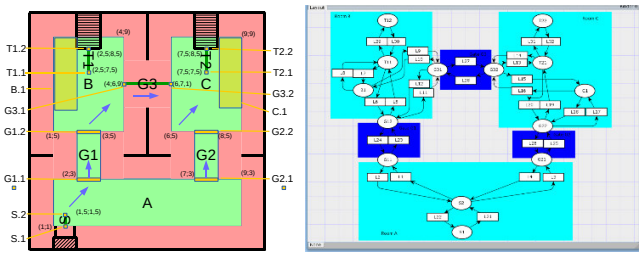

Fig. 1: The layout of the AGVS and its CPN representation.

After specifying the environment, the Petri net must be completed with the properties of the AGVs and their associated routes, and conditions describing the movement of the AGVs between the specific locations in the layout. The color set declarations used to identify AGVs and describe the routes to be taken are shown in Fig. 2.

```
colset PLACE = with S1|S2|G11|G12|G21|G22|G31|G32|B1|T11|T12|
C1|T21|T22|Wh|Delivery|none;
colset ID = INT;
colset PACK_ID = STRING;
colset PACK = product PACK_ID * PLACE;
colset PACK_FROM_TO = product PACK_ID * PLACE * PLACE;
colset PACKLIST = list PACK_FROM_TO;
colset PATH = list PLACE;
colset FORK = with no|ahead|behind;
colset AGV = product ID * PACK * PATH * PACKLIST * FORK * PLACE * PLACE;
```

Fig. 2: The color set declarations of the AGVS

Accordingly, an AGV can be specified by a combination of an identifier (*ID*), the object on the AGV (*PACK*), a route to be completed (*PATH*), the objects to be transported (*PACKLIST*), the position of the fork if exists *(FORK)*, the current and previous location of the AGV *(PLACE)*. *PACK* is described by specifying its *ID* and the destination of the objects, *PATH* contains the list of locations in the order of the route to be taken, and *PACKLIST* contains the *ID*, pick-up station, and delivery station of the objects to be transported by the AGV. An example of specifying the properties associated with an AGV based on the above declarations is: *(2, ("",none), [S2,G11,G12,T11,T12,T11,B1,G31,G32,T21, T22],[("pack1",Wh,Delivery)],behind,S1,none*). This means that the AGV with *ID* 2 is empty, its route to be completed is via the points *S2, G11, ..., T22* in the order given in the list, and during the tour it must transport the object with *ID "pack1"* from the pick-up point *Wh* to the delivery point *Delivery*. The AGV has fork and it is in a behind state (it is positioned at the back according to the direction), the current position of the AGV is *S1* and its previous position is irrelevant.

If the AGV with *ID* 2 is located at the marked point *S1* (charging station), i.e., the above value is assigned to the location *S1* in CPN, the AGV is assigned the route as shown in Fig. 3. The AGV is travelling empty on the red and purple sections of the route and is delivering an object on the blue and purple sections of the route (the AGV is travelling both empty and delivering an object at the purple points of interest). The green sections represent the pick-up and drop-off of the object.

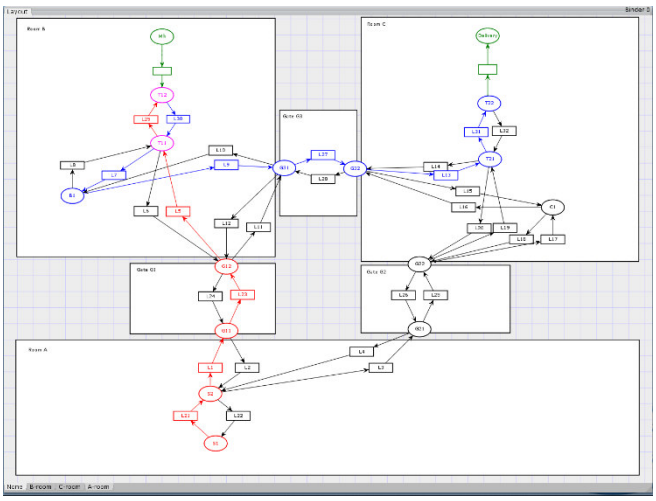

Fig. 3: Access route assigned to the AGV.

To specify the movement of an AGV, the transitions of the CPN require the definition of condition and consequence functions. To perform a transition *T* representing a step of an AGV, i.e., to move the AGV from a given place *P1* to a given place *P2*, it is necessary that

1. the AGV is at place *P1* and
2. the first element of the AGV's route list must be exactly place *P2*.

Of the firing conditions for a transition *T*, condition 1 is given by arc expression function associated with the input arc of the transition and condition 2 is given by guard function associated with the transition.

The consequence of firing transition *T* is that

1. the AGV is positioned at place *P2* and
2. place *P2* is removed from the AGV's route list.

The description of the above conditions and consequences using CPN functions is shown in Fig. 4.

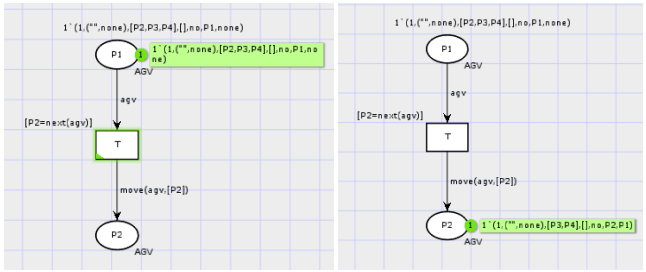

Fig. 4: Transition describing the movement of the AGV with pre- and post-firing states.

In addition to the transitions describing the movement of AGVs, transitions describing the handling of transported objects (pick-up, drop-off) are also required.

To implement the *Pick_up* transition in CPN, i.e., the loading of an object onto an AGV, requires

1. the AGV is at the pick-up point (*P*),
2. the AGV is empty and
3. the storage associated with *P* contains the object that is included in the list of objects to be transported by the AGV.

Of the firing conditions for the *Pick_up* transition, condition 1 can be specified by arc expression function associated with the input arc of the transition, and conditions 2 and 3 can be specified by the guard function associated with the transition.

The consequence of firing the *Pick_up* transition is

1. the object to be transported is deleted from the container associated with the transition and
2. the object to be transported is placed on the AGV and
3. the object to be transported is deleted from the list of further packages to be transported of the AGV.

The conditions and consequences for picking up an object by means of functions assigned to the arcs of CPN are shown in Fig. 5.

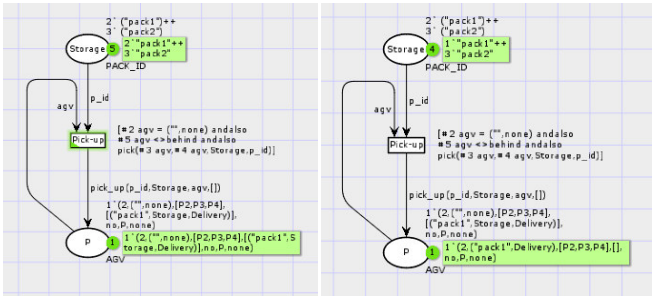

Fig. 5: Transition describing pick-up of object with pre- and post-firing states.

To implement an activity representing the delivery of an object (*Drop* transition) with CPN, the preconditions are as follows:

1. the AGV is at the delivery point (*P*) and
2. the place of arrival of the object transported by the AGV is the container of *P*.

Both firing conditions of the *Drop* transition can be specified by arc expression function associated with the input arcs of the transition.

The consequence of firing the *Drop* transition is that

1. the object to be delivered is removed from the *PACKLIST* of AGV and
2. the object to be transported appears on the container of *P*.

The preconditions and consequences of a *Drop* transition are described by functions assigned to the arcs as shown in Fig. 6.

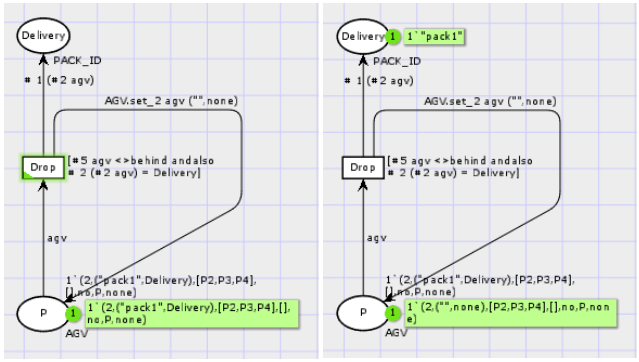

Fig. 6: Transition describing delivery of object with pre- and post-firing states.

Examining the AGVS, it is important to describe the orientation of the AGV and how it changes direction, which also allows questions to be asked about when and how a change of direction is possible without moving the AGV's center of gravity. When managing the orientation of AGV, there are basically two types. One type of AGV does not have a fork, in which case the lifting and unloading of objects is done by moving the AGV under the pallet and then raising and lowering the AGV. This is the operation of the SEW AGV shown on the left in Fig. 7. The other type of AGV is equipped with a fork (lifting arm) for transporting objects. Example is Linde AGV shown on the right in Fig. 7. When modelling these AGVs, it is also necessary to specify whether the AGV travels with a fork in front or behind it on the given route section.

Fig. 7: AGV without fork (left side) and with fork (right side).

In relation to the possible change in the direction of movement of the AGV, it is necessary to identify the places where it is possible to move in the opposite direction. In addition, the points of interest in the layout where the change of orientation of the AGV is mandatory in the event of a further movement shall be identified. Such places are pick-up points, delivery points, charging points and walls. These movements are shown by pairs of steps in different colors in Fig. 8 and 9.

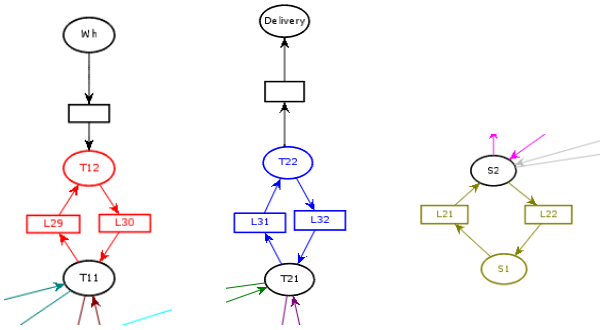

Fig. 8: Orientation change: at the pick-up point (*T12*), at the delivery point (*T22*), at the charging point (*S1*).

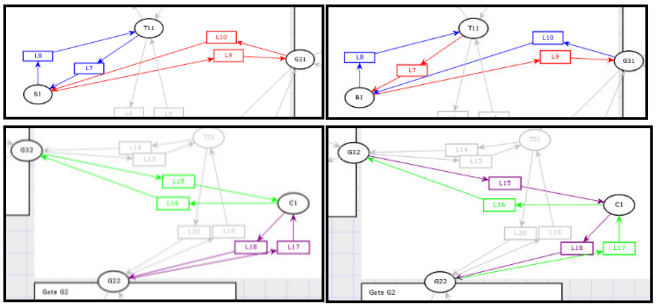

Fig. 9: Orientation change at the walls (*B1* and *C1*).

In addition, depending on the route section to be traversed, mandatory orientation change may also occur due to the direction of the trajectory arc associated with the route section. These locations in the layout are shown in Fig. 10., where at the gateways associated with locations *B* and *C* (marked points *G12*, *G31*, *G32* and *G22*), opposite directional movement is possible according to the route sections to be accessed. These movements are shown by the pairs of steps in different colors in the figure.

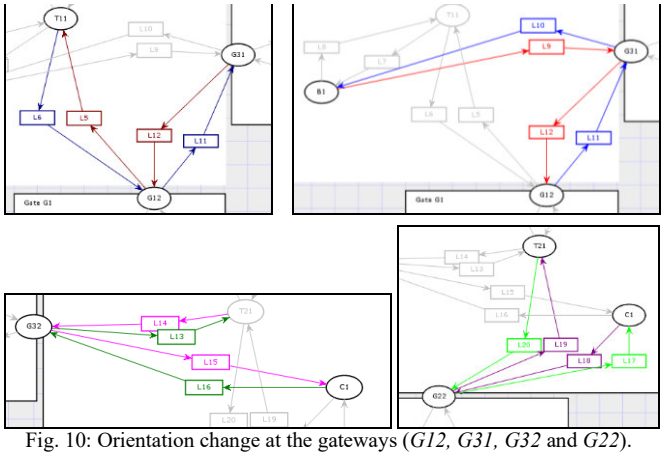

Fig. 10: Orientation change at the gateways (*G12, G31, G32* and *G22*).

Based on the above, the change in orientation of AGVs can be summarized as follows:

- Orientation change required (mandatory, regardless of the road section to be covered)
  - proceeding from a delivery point,
  - proceeding from a pick-up point,
  - proceeding from a charging point
  - passing from a wall.

The required change of orientation depends on the type of the actual location.

- Change of orientation possible (optional, determined by the route to be followed)
  - previous location and next location are the same (change of vehicle orientation due to "turning"),
  - other orientation change, due to the direction of the path of the route section.

The possible change of orientation is influenced by the previous position of the route travelled and the current step. The change in orientation of the AGVs during each step (transition) is implemented in the arc expression functions and guard functions associated with the transitions. (Comment: the capacity limits of the places represented by anti-places of the CPN.)

B. The simulation of AGVS with CPN

The following case study shows the simulation of the colored Petri net model. Fig. 11 shows the initial state of the system: one AGV at location *S1* (charging point of AGV): *(1, ("",none),[S2,G11,G12,T11,T12,T11,B1,G31,G32,T21,T22], ("pack1",Wh,Delivery)],behind,S1,none)*. That is, the AGV with ID *1* is empty, the route to be taken *is [S2,G11,G12,T11, T12,T11,B1,G31,G32,T21,T22]*, the list of the object to be delivered is *[("pack1",Wh,Delivery)]*, the fork is in the *behind* state (backwards according to the direction of the alignment to the charging point), the current location of the AGV is *S1*, and the previous location is irrelevant.

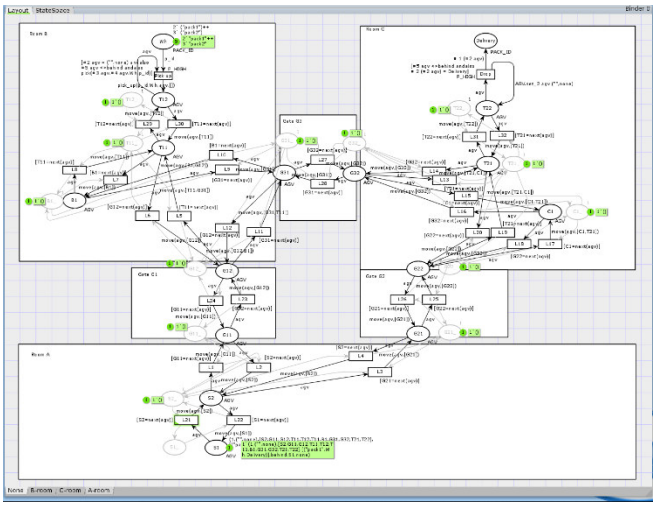

Fig. 11: The initial state of the AGVS.

In the initial state shown in Fig. 11., transition *L21* can be performed, after which the system is in the state shown in Fig. 12. (AGV with ID *1* moves to the state *S2*, its orientation changes to *ahead*, i.e., the AGV is positioned in the front according to the direction of movement).

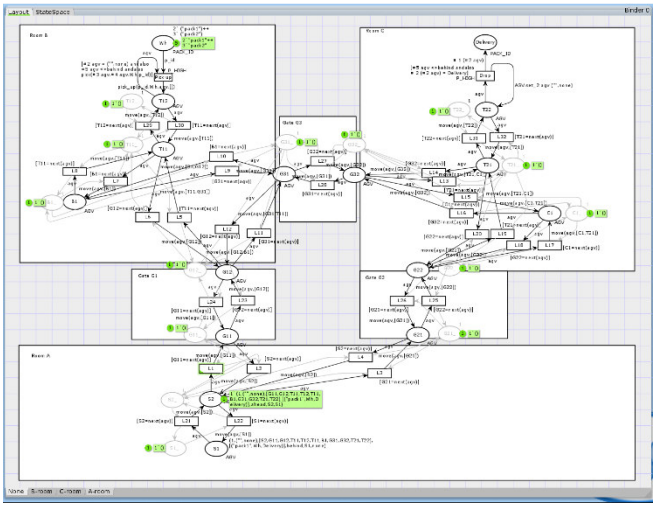

Fig. 12: The state of the AGVS after firing *L21 – L1* transition is fireable.

In the state shown in Fig. 12, transition *L1* is firing, the AGV is moved from the marked position *S2* to the marked position *G11* after the transition is executed. Subsequent firing transitions are *L23, L5, L29*, which, after successive transitions, move the AGV to *T12* marked point in room *B*. The orientation of the forklift remains unchanged (*ahead*) throughout. The AGV at the *T12* location picks up the *"pack1"* object, which must be delivered to the *Delivery* location. The state after the execution of the *Pick-up* transition is shown in Fig. 13.

In the state shown in Fig. 13, transition *L30* is firing, i.e., the AGV can move to the marked position *T11*, during which the AGV orientation changes (becomes *behind*). After executing transition *L30*, transitions *L7* and *L9* are firing. Upon execution of the latter, the orientation of the AGV will change again (become *ahead*). Then, transitions *L27*, *L13* and *L31* will be executed, which will bring the AGV to the marked point *T22* of room *C*, i.e., the delivery point, were, after execution of the *Drop* transition, the system will be in the state shown in Fig. 14.

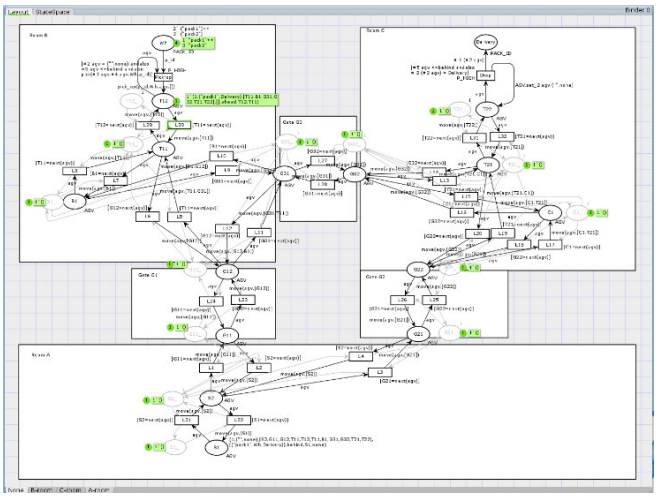

Fig. 13: The state of the AGVS after firing *Pick-up* transition.

The package with the ID *"pack1"* has been dropped from the AGV and at the same time has appeared in the *Delivery* container. In the state shown in Fig. 14., the AGV is in the empty state at the marked position *T22* (delivery point). There is no transition to be executed, therefore the current state is deadlock. The list representing the current route to be taken by the AGV and the list of objects to be delivered are both empty ([]), indicating that the AGV has completed its assigned route and delivered the assigned package(s). Thus, the deadlock reached is also the expected end state of the system.

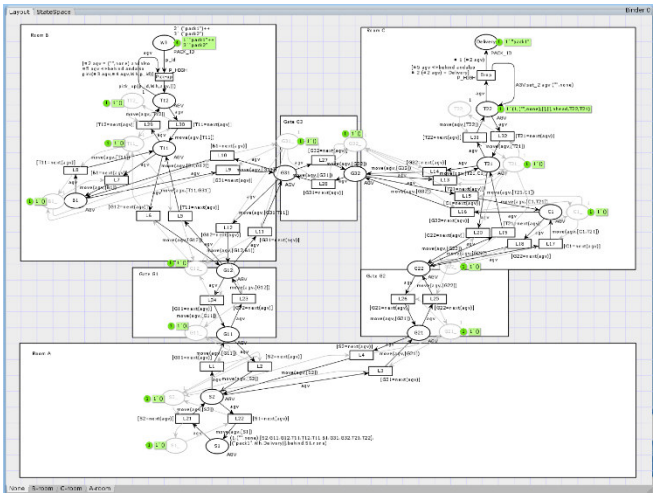

Fig. 14: The deadlock state of the AGVS – expected state.

C. *The analysis of AGVS with CPN*

In addition to simulating the AGVS, the relevant dynamic properties of the system, such as reachability and deadlock, are investigated using the reachability graph [8] with the State Space Tool integrated in CPN Tools.

In the initial state of the case study, there are two AGVs in the environment, as shown in Fig. 15. One of them is at location *T22*, with characteristics *(1,("",none),[T21,G22, G21,S2,S1],[],ahead,T22,none)*. The other AGV is at location *T12*, with characteristics *(2,("",none),[T11,B1,G31, G32,T21,T22] ,("pack1",Wh,Delivery)],ahead,T12,none)*.

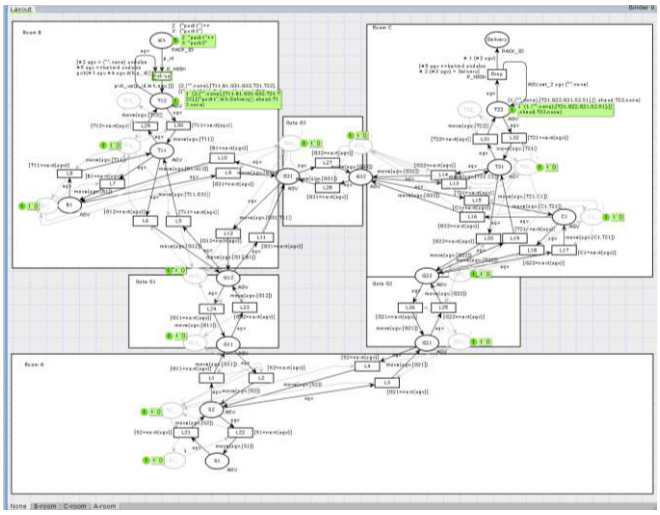

Fig. 15: The initial state of the AGVS with two AGVs.

The reachability graph of the system is shown in Fig. 16. State *1* of the graph represents the initial state, states *21* and *44* represent deadlocks, and their corresponding states values are given in the rectangles.

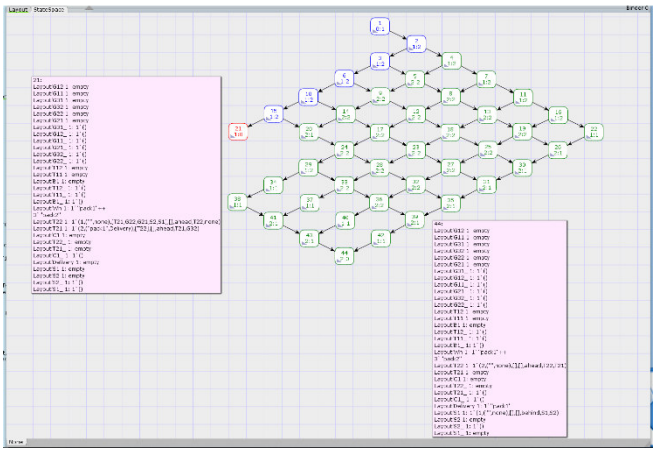

Fig. 16: The reachability graph of the AGVS with two AGVs.

In state *44*, AGV *1* is located at *S1* (charging station) and AGV *2* at *T22* (delivery station). Both AGVs have an empty ([]) remaining route to follow and package list to deliver, i.e., they have traversed their assigned route and delivered their assigned objects. The deadlock is the intended end state of the system. In state *21*, AGV *1* is at *T22* and AGV *2* is at *T21*. The properties of AGV *1* are the same as in its initial state, i.e., the AGV is 'stationary'. AGV *2* has travelled all but one place along the designated route but can no longer enter place *T22*. The state is a deadlock, which is the unexpected final state of the system.

The partitions of the reachability graph are shown in different colors in Fig. 16.: the states in the red partition whose result in unexpected final state(s) regardless of the execution order of subsequent transitions. The states in the green partition show the states that lead to an expected final state. The states in the blue partition are the ones that can lead to both types of final states.

If the orientation of the AGV with ID *2* in the initial state is changed while leaving the other properties unchanged (i.e., the color of the token representing the AGV at location *T12* is *(2, ("",none), [T11,B1,G31,G32,T21,T22], [("pack1",Wh, Delivery)],behind,T12,none)*), the reachability graph shown in Fig. 17. is obtained.

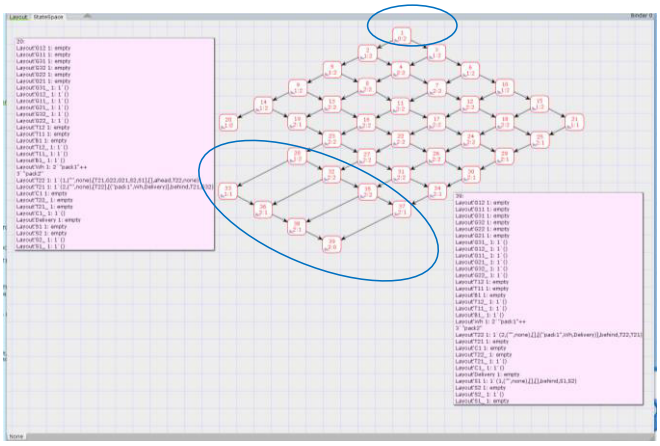

Fig. 17: The reachability graph of the AGVS with two AGVs (modified orientation of AGV *2* in the initial state).

Comparing this with the reachability graph in Fig. 16., it can be seen, that the number of vertices and edges of the graph has decreased (i.e., previously feasible transitions have become infeasible), and therefore the structure of the reachability graph has naturally changed (the location of the changes is shown by blue ovals). Both deadlocks of the system are not expected system states, and therefore there is no transition sequence leading to an expected system state (each vertex belongs to a "red partition").

IV. Conclusions

This paper presents the application of colored Petri nets to the description and analysis of automated guided vehicle (AGV) systems, showing that the colored Petri net model is suitable for system simulation and formal analysis. The analysis of the Petri net reachability graph has shown that for the design and testing of AGVS, it is critical to identify deadlocks and to avoid them by detecting unexpected system states as soon as possible to determine alternative step sequences.

V. References

[1] H. Kaid, A .M. El-Tamimi, E. A. Nasr, A. Al-Ahmari, “Applications of Petri nets based models in manufacturing systems: A review” Proceedings of the International Conference on Operations Excellence & Service Engineering, Orlando, FL, pp. 516-28, 2015.

[2] W. Wu, Z. Xing, H. Yue, H.. Su, H., S. Pang, “Petri-net-based deadlock detection and recovery for control of interacting equipment in automated container terminals” IET Intelligent Transport Systems, 2022.

[3] C. A. Petri, “Kommunikation mit Automaten”, Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn, Bonn, Germany, 1963.

[4] J. L. Peterson, “Petri Net Theory and the Modeling of Systems”, Prentice Hall, New Jersey. 1981.

[5] K. Jensen and L.M. Kristensen, “Coloured Petri Nets. Modelling and Validation of Concurrent Systems”, Springer-Verlag 2009.

[6] R. Lakner and B. Bertók, “Synthesis and analysis of AGV systems”, AIS 2020, 15th International Symposium on Applied Informatics and Related Areas, Óbuda University, pp. 63-67., ISBN 978-963-449-209-2, 2002.

[7] CPN Tools: A tool for editing, simulating, and analyzing Colored Petri nets, cpntools.org, Last accessed 01/10/2022.

[8] T. Murata, “Petri Nets: properties, analysis and applications”, In: Proc IEEE, vol. 77. pp. 541-80. 1989.