Journal of Intelligent Manufacturing (1992) **3**, 379–390

Automated guided vehicle systems and their Petri-net properties

SUHUA HSIEH and YING-JER SHIH

Department of Mechanical Engineering, National Taiwan University, Taipei, Taiwan 10764, ROC

Received September 1991 and accepted March 1992

This paper defines clearly and presents the fundamental properties of automated guided vehicle systems (AGVS) and their relationship to Petri nets. The objective is to develop an intelligent modelling tool to allow the design of robust AGVS models using some invariant Petri-net properties. The Petri-net properties of the system are presented in terms of safeness, boundedness, strict conservation, reachability and liveness. The robustness of the system at floor-level is embedded into the model by introducing system traffic collision-free, constant numbers of vehicles and traffic control signals, path reachability, and system traffic deadlock-free. Two different merge structures are presented to illustrate the effectiveness of the present approach. The result shows the strength of Petri-net theory in the modelling of complex AGVS. The AGVS models thus built are robust.

Keywords: AGVS, Petri-net properties, robust, union

1. Introduction

A desirable system modelling tool is to provide not only the intelligence but also the robustness of the system. The greater intelligence the tool provides, the less modification the model needs. The system development cycle time is significantly reduced if a versatile modelling tool is used. The utilization of intelligent and robust modelling tools is the key to the success of system design. AutoMod (1989) and MAP/1 (Miner and Rolston, 1987) are powerful tools for the modelling of manufacturing systems. Unlike other general-purpose modelling tools, a machine tool in AutoMod is not just a symbol. It is a device which can machine parts. Whenever the user generates a machine tool, the related attributes regarding machine tools are also created. The necessary system conditions and properties are provided when the model is built. However, most of these existing modelling tools are simulation-oriented and only good for special systems. In an attempt to develop a versatile and rather general modelling tool for complex automated guided vehicle systems (AGVS), this study investigates the necessary system conditions and properties needed for the development of a robust modelling environment.

If the AGVS is adopted as the major material handling system in an automated manufacturing plant, all manufacturing workstations and storage/retrieval systems will have to be integrated together physically. Any activity in the system will affect directly or indirectly the performance of the AGVS. These activities need to be taken into consideration in the system design process. A powerful yet flexible design tool becomes very essential. Unfortunately, many existing modelling tools, such as AutoMod, are not design-oriented. The design is usually refined by repeating the same modelling sequences. The momentum to develop a design tool is somehow slowed down in the USA due to the lack of success in many existing AGVS. Although many companies are reluctant to invest in the design and development of AGVS at present a report in Modern Material Handling, (1990) indicated that the AGVS remains as one of the most flexible material handling systems for industrial automation. Japanese corporations are reported to have purchased about 5000 vehicles. European companies bought an estimated 3000 vehicles in 1989 (a report in Modern Material Handling, 1990). Therefore, the development of a powerful AGVS design-oriented modelling tool is expected to remain in high demand in the near future.

The Petri-net theory is known as a versatile tool for modelling systems with interacting concurrent components (Kamath and Viswanadham, 1986; Martinez *et al.*, 1986; Crockett *et al.*, 1987). Many advantages in using Petri nets for modelling complex systems have been reported. These advantages include:

(1) The system models built upon Petri-net theory provide better physical insight into the system since they are presented graphically;
(2) A systematically qualitative analysis of the system model is feasible with the existence of many well-developed Petri-net techniques (Peterson, 1981);
(3) Off-line system simulation can be done to evaluate the system and validate the design;
(4) The simulation model can be used directly as the on-line system control model;
(5) The performance evaluation of the system is possible by utilizing timed Petri nets;
(6) Any Petri-net model can be treated as a submodel of another Petri-net model; in other words, Petri nets can be used to model a system hierarchically.

Because of these advantages, the Petri net is adopted as the basic tool for the development of AGVS models in this paper.

Hsieh and Lin (1991; 1992) presented in their earlier work some basic Petri subnets such as line, divide, merge and intersection nets for AGVS using Petri nets as the modelling language. Their subsequent work utilized the union of these basic subnets to construct an AGVS model. Although the background work for establishing the model step by step has been established, the detailed implementation procedures have not yet been addressed so far. In order to develop such a modelling tool, we define clearly the characteristics of the AGVS systems and Petri-net properties in this paper. The attempt is to develop some elementary subnets needed for the construction of robust AGVS models.

2. Petri-net theory

Petri-net theory is a useful tool for the study of complex systems with interacting concurrent components (Peterson, 1981). Petri-net theory allows a system to be modelled by a Petri net, or a so-called mathematical representation of the system. Important information about the basic structure and the dynamic behaviour of the system are characterized by a net model. The model can then be used for system evaluation and/or for design modifications.

2.1. *Mathematical structure*

Definition 1: A Petri-net structure, C, is a four-tuple, $C = (P, T, I, O)$ where:

(1) $P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of places, $n \geq 0$;
(2) $T = \{t_1, t_2, \ldots, t_m\}$ is a finite set of transitions, $m \geq 0$. The set of places and the set of transitions are disjoint, $P \cap T = \varnothing$;
(3) $I: T \rightarrow P^\infty$ is the input function, a mapping from transitions to bags of places;
(4) $O: T \rightarrow P^\infty$ is the output function, a mapping from transitions to bags of places.

A bag is a generalization of sets which allows multiple occurrences of an element in a bag.

A Petri net is usually represented by an oriented bipartite graph with places represented by circles, ○, and transitions represented by bars, |. There is an arc joining a place p_i to a transition t_j iff $\#(p_i, I(t_j)) \neq 0$. Analogously, there is an arc from a transition t_j to a place p_k iff $\#(p_k, O(t_j)) \neq 0$. Natural numbers, $\#(p_i, I(t_j))$ and $\#(p_k, O(t_j))$, are called the weights of the arcs.

Definition 2: A marking μ of a Petri net is a function $\mu: P \rightarrow N$. It gives the number of tokens contained in each place $p \in P$. The marking μ can also be defined as an n-vector, $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$, where $n = |P|$ and each $\mu_i \in N$, $i = 1, \ldots, n$. The vector μ gives for each place p_i in a Petri net the number of tokens in that place. A token is usually represented by a dot. The meaning of the marking represents the state of a Petri net. The execution of a Petri net is done by firing transitions. A transition is fired by removing tokens from its input places and creating new tokens which are distributed to its output places.

Definition 3: A transition $t_j \in T$ in a marked Petri net $C = (P, T, I, O)$ with marking μ is enabled if for all $p_i \in P$,

$$\mu(p_i) \geq \#(p_i, I(t_j))$$

Definition 4: A transition t_j in a marked Petri net with marking μ may fire whenever it is enabled. Firing an enabled transition t_j results in a new marking μ' defined by

$$\mu'(p_i) = \mu(p_i) - \#(p_i, I(t_j)) + \#(p_i, O(t_j))$$

and μ' is immediately reachable from μ.

Definition 5: The change in state from μ to μ' caused by firing a transition t_j may be defined by a change function δ called the next-state function such that $\delta(\mu, t_j) = \mu'$. For a sequence of transitions $\sigma = (t_{j_1}, t_{j_2}, \ldots, t_{j_k})$, the marking $\mu' = \delta(\mu, t_{j_1}, t_{j_2}, \ldots, t_{j_k})$ is the result of firing the sequence of transitions σ.

Definition 6: A directed arc from a place p_i to a transition t_j has a small circle rather than an arrowhead at the transition. This directed arc is named *inhibitor arc*. The small circle means 'not'. The firing rule is changed as follows: a transition is enabled when tokens are in all of its normal inputs and zero tokens are in all of its inhibitor inputs; the transition fires by removing tokens from all of its normal inputs.

2.2. *Net properties*

Petri nets are capable of characterizing a large variety of systems. The model built based on Petri nets can be used to analyse the behaviour of the system. The analysis of the Petri-net model can lead to a better understanding of system insights. Mathematically speaking, there exist some fundamental properties of a net that embed the robustness into the system. These properties include safeness, boundedness, conservation, and liveness.

(1) *Safeness*: a place in a Petri net is safe if the number of tokens in the place never exceeds one. A Petri net is safe if all places in the net are safe;

(2) *Boundedness*: a place in a Petri net is bounded if the number of tokens in the place never exceeds k. It is said that the place is k-safe or k-bounded. A Petri net is k-safe if every place of the net is k-safe;

(3) *Conservation*: a Petri net with initial marking is conservative with respect to a weighting vector W, $W = (w_1, w_2, \ldots, w_n)$, $n = |P|$, $w_i > 0$, if for all $\mu' \in R(C, \mu)$,

$$\Sigma_i w_i \cdot \mu'(p_i) = \Sigma_i w_i \cdot \mu(p_i)$$

Note: $R(C, \mu)$ is the reachability set of the net. If every element in the weighting vector is 1, or $W = (1, 1, \ldots, 1)$, the Petri net is referred to as a strictly conservative net. To determine a positive weighting vector with respect to a conservative net, the net must be bounded (Agerwala, 1979; Peterson, 1981; Jensen, 1987). Hence, if the net is conservative, the net must be bounded;

(4) *Liveness*: a transition is live in a marking μ if it is potentially fireable in every marking in $R(C, \mu)$. There are four different levels of liveness. The liveness used in this paper is the level 4 liveness, which is defined as: a transition t_j is live at level 4 if for each $\mu' \in R(C, \mu)$ there exists a firing sequence σ such that t_j is enabled in $\delta(\mu', \sigma)$.

(5) *Reachability problem*: given a Petri net C with initial marking μ and a marking μ', if there exists a $\mu'' \in R(C, \mu)$ such that $\mu'' \geq \mu'$, it becomes a reachable problem. Peterson (1981) proved that the reachability problem is equivalent to the liveness problem.

2.3. *Modelling techniques*

Two major Petri-net modelling techniques, the reachability tree and the incident matrix equations, are presented. These techniques provide problem-solving mechanisms for some problems discussed above. The boundedness, safeness and conservation properties of a net can be obtained either by analysing the reachability tree or the incident matrix equation. The incident matrix equation provides merely necessary conditions to solve the liveness or reachability problems. The reachability tree can be used to solve the liveness problem for those nets having a sequence of transition firings repeatedly.

2.3.1. *The reachability tree*

The reachability tree represents the reachability set of a Petri net. Every reachable marking will be produced eventually. It may be infinite. However, each node i in the tree is associated with an extended marking $\mu(i)$. Each node is also classified as either a frontier node, a terminal node, a duplicate node, or an internal node.

By observing the reachability tree, one can determine the safeness, boundedness and conservation properties of the Petri net. A Petri net is bounded if and only if the number of reachable markings in every place is finite. If the bound for all places is 1, then the net is safe. Conservation can be tested effectively using the reachability tree. If the weighted sums for each reachable marking are the same, the net is conservative with respect to the given weight. Although the reachability tree does not necessarily contain enough information to always solve the problems of liveness and reachability, the tree may have sufficient information to solve many such problems. For example, a net whose reachability tree has a terminal node is not live. Similarly, a marking μ' of a reachability problem may appear in the reachability tree, and if so it is reachable. By observing a net, one can determine a live net if nodes in every separate subtree are associated with markings which are the same as that of the frontier node, and every transition has been fired at least once. This observation is demonstrated later (see Fig. 6).

2.3.2. *Incident matrix equation*

An alternative to the (P, T, I, O) definition of Petri nets is to define two matrices N^- and N^+ to represent the input and output functions. Each matrix is m rows (one for each transition) by n columns (one for each place). We define $N^-[j, i] = \#(p_i, I(t_j))$ and $N^+[j, i] = \#(p_i, O(t_j))$. The difference between N^+ and N^- is defined as the incident matrix.

When considering a sequence of transition firings σ, let μ' be the new marking after the sequence of firings, and μ be the marking before firings.

$$\mu' = \delta(\mu, \sigma) = \mu + f(\sigma) \cdot N$$

where $f(\sigma)$ is a 1 by m firing vector. When a conservation problem is considered, the problem definition can be stated as: given a marked Petri net, the Petri net is defined as a conservative net if and only if there exists a positive weighting vector W such that $N \cdot W = 0$. If the net is conservative and every element of the weighting vector is 1, the Petri net is strictly conservative.

3. AGVS properties

An AGVS consists of three parts – vehicles, flow paths and system management. The main function of vehicles is to move parts from one station to another. As long as part transportations can be carried out, the vehicle does not require much wisdom. Generally speaking, vehicles are passive and under the control of the system controller. The system management makes most of the decisions such as what is the next task and which vehicle will be assigned. Commands determined by the system controller are issued through wires if it is a wiring system to vehicles, and the vehicle status may also be reported through wires to the controller. As for how the system management performs tasks, it depends on the types of applications. The details of this issue were addressed earlier by Hsieh and Lin (1991).

The flow-path design is important for an AGVS. As pointed out by Hsieh and Lin (1991), a well-designed AGVS mechanism simplifies management problems. For instance, if the vehicle work area is divided into several different portions, the job dispatch and vehicle routeing problems become less difficult. If the floor-level control is capable of preventing vehicles from collision, the system management can concentrate on other system problems. Due to the poor layout of the flow paths, the traffic load of a particular area of the system is heavier than others, a traffic jam may easily occur in that area, and the workload of the system management will be increased.

An AGVS has three levels of robustness. Level 1 is the vehicle-level robustness, level 2 is the floor-level robustness, and level 3 is the management-level robustness. The vehicle-level robustness depends upon the vehicle manufacturer. The management-level robustness has a lot to do with types of applications, company policies and future expansions, and is case-dependent. The floor-level robustness has two phases. One is from the plant layout to the design of a robust flow-path layout, and the other is from a flow-path layout to the construction of a robust flow-path system. A good design of flow-path layout is case-dependent. One has to take workstations, production processes, workpart routeings, etc. into consideration. However, once the flow-path layout is determined, whatever is left is common to all applications. Since the floor-level design is important, the design of the common part for the AGVS user should promote the use of AGVS. Hence, the determination of necessary properties for a robust AGVS at the floor level is worthy of study.

3.1. *Floor-level robustness properties*

The floor-level robustness is one of three levels of AGVS robustness. The floor-level robustness is addressed in terms of a system traffic collision-free concept, this assumes that there are constant numbers of vehicles and traffic control signals, path reachability, and that the system traffic is deadlock-free. In order to build a robust flow-path system, these properties need to be built into the system with a robust flow-path layout. To implement the zone control functions systematically by Petri nets, zones are divided into physical zones, pseudo zones and composite pseudo zones. The details of each properties and zone functions are discussed below.

The invariant properties of a robust AGVS at the floor level are the key to the design of an AGVS. In many applications, zone control is used to prevent vehicles from collision. Obviously, being collision-free is one of the important AGVS properties at the floor level. Vehicles served in the system are expected to move under predictable commands within the specified area. This indicates that the flow path and the traffic control of the AGVS should be a closed loop. Since vehicles served in the system will not disappear, and the traffic control signals are always there waiting for the next incoming vehicles, the total number of vehicles in the system is a constant, and the total numbers of traffic control signals in each control area and in the whole system are constants. From a cost viewpoint, since every path requires costs for both construction and maintenance, it is economic and convenient to have every path being reached and used repeatedly. In other words, paths in the system are always reachable. Therefore, path reachability is another important system property.

In normal system operation, if three or more than three vehicles are locked by themselves, and any vehicle can move only if any one of the other vehicles can move, the traffic system is dead. The situation is generally referred to as system deadlock. A system which is deadlock-free is essential, and therefore, it is an important system property. In conclusion, the design of a robust AGVS should include the following properties:

(1) System traffic collision-free;
(2) A constant number of vehicles in the system;
(3) Constant numbers of traffic control signals in every control area and in the whole system;
(4) Path reachability;
(5) System traffic deadlock-free.

It is difficult to solve system traffic blocking problems without a time factor at this level (Hsieh and Lin, 1992). We will address only the five aspects mentioned above of floor-level robustness.

3.2. *Floor-level traffic zones*

Since zone control is widely used to prevent vehicles from collision, we adopt zone control as a technique for traffic modelling. To be able to implement zone control funtions by Petri nets, zones are distinguished in this paper by physical zones, pseudo zones and composite pseudo zones.

(1) *Physical zone.* The zone is usually referred to as a segment of the flow path that allows vehicles to stop one at a time. The zone does not have a specified length as long as a vehicle can be held in it. The solid-line-frame square is used to represent the physical zone in the flow-path layout. This is shown later in Fig. 1.

(2) *Pseudo zone.* The zone covers the portions of the intersection of two paths and is near the intersection. Unlike the physical zone, the pseudo zone does not need to have enough length to hold a vehicle. The pseudo zone is used as a control area where more than one vehicle passing through the zone at a time is not allowed. The vehicle that arrives at any pseudo zone is not allowed to stop or do any jobs; it has to leave immediately. Therefore, any vehicle that is intended to enter a pseudo zone is requested to make sure that

(i) No other vehicle is or is going to be at the pseudo zone;

(ii) The physical zone immediately after the pseudo zone is free, so that the vehicle can leave the pseudo zone right away.

A dash-line-frame square is used to represent the pseudo zone in the flow-path layout. This is also shown later in Fig. 1.

(3) *Composite pseudo zone.* The zone covers the portions of the intersection of several (more than two) paths and near the intersection. The function of the composite pseudo zone is the same as that of the pseudo zone. The traffic control method and the graphical representation of that is the same as those in the pseudo zone.

4. AGVS and petri nets

The robustness properties of AGVS are discussed above. Four fundamental issues need to be addressed before the AGVS Petri-net model of Hsieh and Lin (1992) can be built. These fundamental issues to be addressed include:

(1) What is the Petri-net-based AGVS structure?

(2) What are the important properties of Petri-net-based AGVS models?

(3) How to embed those important properties into models?

(4) How to identify the net properties?

4.1. *AGVS Petri-net structure*

A Petri-net structure is composed of places, transitions and directed arcs. The execution of a net is achieved by firing transitions. Since Petri-net theory is chosen as the modelling tool, the meanings of Petri-net representations such as tokens, places, and transitions need to be addressed. By adding some control nodes to a plain net, Hsieh and Lin (1992) constructed several AGVS traffic control nets. The plain net reflects the layout of the flow paths. The control nodes prevent vehicles from having accidents, which can be implemented physically by the use of electric relay switches and writing the necessary Boolean functions into the AGVS controller. A similar method is used here to establish the AGVS net. The net is composed of two major portions, the flow-path net and the control-loop net. The flow-path net reflects the layout of the flow paths, and the control-loop net performs the traffic control function. These two nets and the components of the two nets are described below.

4.1.1. *Flow-path net*

Each place in the net may represent one of the three zones. The direction of traffic flow is represented by a directed arc. Tokens in places represent vehicles. Transitions represent permissions for vehicles to move from one place to another place. When a transition is fired, the vehicle is permitted to proceed to the place directed by the arc. The permission is issued by the controller whenever the request is made and the firing rules are satisfied. To fulfil the zone control function, the transition firing rules are modified and stated as follows:

Definition 7: A transition $t_j \in T$ in a marked AGVS Petri net $C = (P, T, I, O)$ with marking μ is enabled if for all $p_i \in P$, $\mu(p_i) \geq \#(p_i, I(t_j))$ and $\mu(O(t_j)) = 0$, and an enabled transition may fire upon request.

4.1.2. *Control-loop net*

Each place in this net represents a control node which contains some electric components such as wires, sensors, relay switches etc. Wires or sensors generate input signals to report the current status to the controller. Relay switches produce output signals or commands from the controller to supervise vehicles. Tokens in each place represent signals or commands. Directed arcs are used to direct the flow of signals or commands. There is no other set of transitions which belong to the control-loop net itself. Both the control-loop net and the flow-path net share the same transitions. Because of that, the union of the two nets becomes trivial. One net is then put on to another as long as the transitions are matched. Examples are given below to illustrate the effectiveness of the present method.

4.2. *AGVS Petri-net properties*

A robust AGVS Petri-net structure is composed of a flow-path net and a control-loop net. The AGVS model with built-in Petri-net properties possesses safeness, boundedness, strict conservation, reachability and liveness. These properties are addressed below.

4.2.1. *System traffic collision-free*

It is essential that a system is designed to be collision-free. This can be achieved by two different methods. On-board vehicle sensing is one way to prevent collision. However, if the vehicle speed is too fast for it to stop, an undesirable traffic accident may occur. The zone control is another method. As mentioned above, the AGVS basic net can be divided into two parts, the flow-path net and the control-loop net. Tokens in the flow-path net represent vehicles, and tokens in the control-loop net represent traffic control signals or commands. According to the zone control rule, one vehicle is allowed in a zone at a time; the flow-path net must be Petri-net safe. Since signals or commands are generated by the electric components, and the total number of components is limited in each zone, the number of signals in a place is limited. The control-loop net must be Petri-net bounded. Therefore, a collision-free system can be obtained automatically by incorporating into the AGVS model the Petri-net safeness property in the flow-path net, and the boundness property in the control-loop net.

4.2.2. *Constant numbers of vehicles and traffic control signals*

When the AGVS starts to operate, the total number of vehicles in the system should be a constant, since it is a closed-loop system. This means that the total number of tokens (vehicles) in the flow-path net has to be a constant. From the traffic control point of view, if the traffic control system is in normal operation condition, the control signals in one area cannot be switched to another. Therefore, the total number of tokens (signals) in every control-loop net (area) should be a constant (i.e. the control-loop net is a closed-loop system). Since the total number of tokens in a control-loop net is constant, the total number of control signals in the whole system must be constant too. Hence, both the flow-path net and control-loop net must be Petri-net strictly conservative in order to satisfy this AGVS property. This property guarantees the strict conservation of system resources, no matter how they are allocated.

4.2.3. *Path reachability*

The reachability problem is one of the basic requirements in the AGVS. If a segment of a path cannot be reached from other segments of paths, the segment of the path is redundant. The flow-path layout will have to be modified. For a robust AGVS, paths are expected to be reached from any place. For each path, there exist several possible routes by which the path can be reached. If the current state is known, one of these possible routes can be used to reach the path. This means that the path is reachable. For a Petri-net $C = (P, T, I, O)$ with marking μ, if there exists a sequence of transitions $\sigma \in T$ such that $\delta(\mu, \sigma) = \mu'$, it is said that a marking μ' is reachable.

The path reachability problem is not exactly equivalent to the Petri-net marking reachability problem. However, if a marking which represents a state is reachable, and the state shows that the desired path is reached, then the path is reachable. The marking reachability problem can be used to identify that the path is reachable. Therefore, the reachability property is the corresponding AGVS Petri-net property. Note that, if every path is reachable, there is no deadlock and the system is live. Therefore, as long as one can identify that the system is live, every path is reachable.

4.2.4. *System deadlock-free*

In a Petri-net, a new marking can be reached if the transition is fired. The new marking represents a new AGVS state, and the transition firing represents a vehicle moving from one place to another. Therefore, if one would like vehicles to execute jobs smoothly in the system, every transition must be live at level 4 (Peterson, 1981). This guarantees that the system is live (or deadlock-free) all the time. If a transition cannot be fired (live at level 0), or can be fired only once (live at level 1) or several times (live at level 2), the output place of the transition is not or will not be reachable through this transition now or later on. Hence, the flow-path layout is not robust. This confirms that the liveness at level 4 property is the corresponding AGVS Petri-net property.

5. Case study

Two different merge structures are used to demonstrate the procedures to embed necessary properties into nets, and the process to identify properties that a net owns.

5.1. *Uni-directional merge case study*

A uni-directional merge flow-path layout is given in Fig. 1. There are three physical zones Z_1, Z_2 and Z_4 and one pseudo zone Z_3. The function of a pseudo zone, as mentioned above, is as a control area. The zone in Petri nets can be represented by a flow-path-net place, and will be used as a pivot when the flow-path net is joined with the control-loop net.

5.1.1. *Construction of nets*

The flow-path is constructed using the layout in Fig. 1. The resulting net is given in Fig. 2. In addition to the four places representing the four zones Z_1, Z_2, Z_3 and Z_4, there are three transitions t_1, t_2 and t_3 in the net. For the analysis purpose, tokens that represent vehicles are placed in Z_1 and Z_2 initially. When both vehicles in Z_1 and Z_2 want to move through Z_3 to Z_4, it is possible that they will collide with each other. Obviously, a traffic control system is needed to supervise vehicles entering zone Z_3. Hence, t_1 and t_2 cannot be fired at the same time. By using the logic design concept (Mano, 1979), two control nodes, places C_1 and C_2, are used to control the firings of t_1 and t_2. A control-loop net for this merge case is then built and is presented in Fig. 3. In Fig. 3, place C_1 is used to indicate the availability of zone Z_3, and place C_2 is used to conserve the tokens in the net (since every control-loop net is a closed-looped system). In this case, the desired logic is simple: one can simply use intuition to construct the control-loop net. For those cases which cannot be determined intuitively, the Karnaugh map method (Mano, 1979) is suggested. After the construction of the two nets, the complete net can be obtained by placing the control-loop net over the flow-path net and superimposing C_2 and the pivot Z_3. The result is shown in Fig. 4.

5.1.2. *Flow-path net analysis*

It is essential to understand the system by analysing the net. Since the merge net constructed above is not a closed system, a modified closed-loop flow-path net is needed. The modified net is presented in Fig. 5, where vehicles can be directed to move from place Z_4 to Z_1 or Z_2 by firing t_5 or t_4. The incident matrix of the modified flow-path net N_f can be given as:

$$N_f = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{c} \begin{array}{cccc} Z_1 & Z_2 & Z_3 & Z_4 \end{array} \\ \begin{pmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & 0 & -1 \end{pmatrix} \end{array}$$

There exists a weighting vector $W_f = (1, 1, 1, 1)^T$ such that $N_f \cdot W_f = 0$. Therefore, the flow-path net is strictly

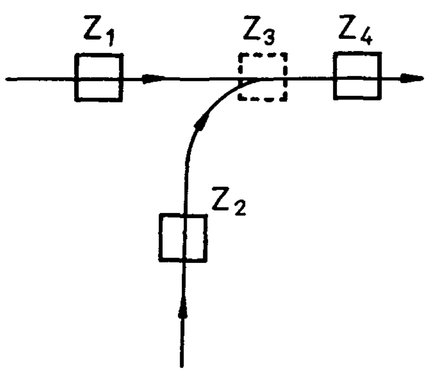

Fig. 1. The uni-directional merge flow-path layout.

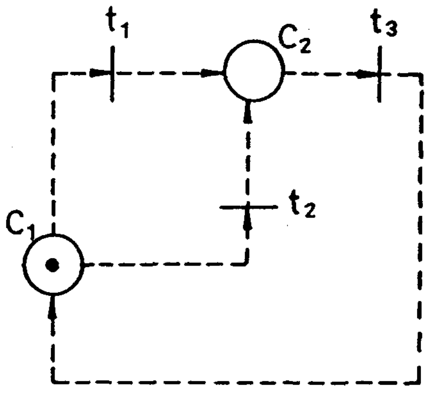

Fig. 3. The control-loop net of the uni-directional merge structure.

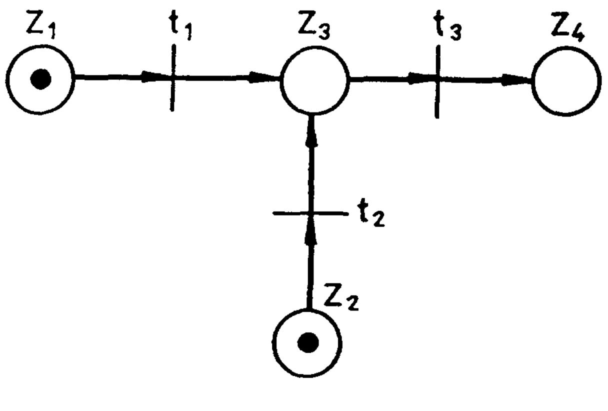

Fig. 2. The flow-path net of the uni-directional merge structure.

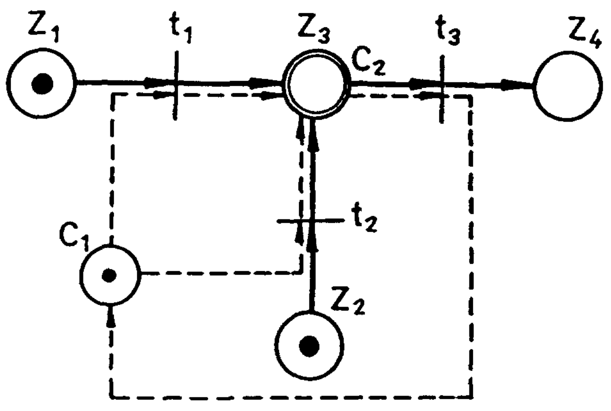

Fig. 4. The uni-directional merge net.

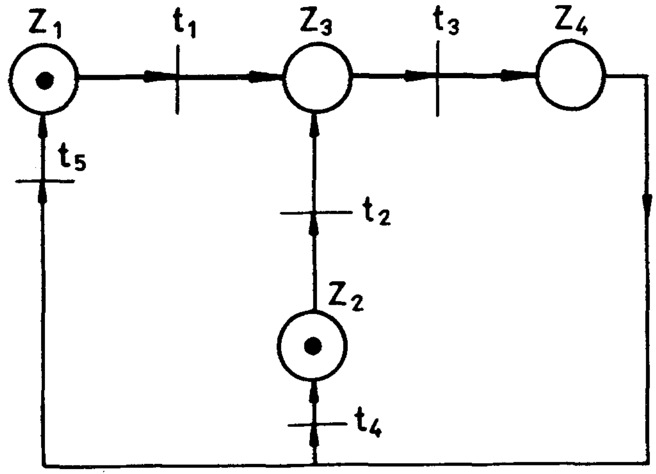

Fig. 5. The modified closed-loop flow-path net of the uni-directional merge structure.

conservative. Because it is conservative, the net is bounded too.

The reachability tree of the flow-path net is presented in Fig. 6. From the figure, one can tell that the net is safe since every reachable marking is safe. As for liveness and reachability, these can also be determined by observing the reachability tree. In Fig. 6, every transition has been fired at least once, and the marking of every subtree returns to the initial marking. The net satisfies liveness and reachability properties. Therefore, the flow-path net is robust.

5.1.3. *Control-loop net analysis*

Because the traffic control signals in every control area will not disappear, the control-loop net is a closed-loop system. The modification of the control-loop net is not necessary in net analysis. The incident matrix of the control-loop, N_c, can be given as:

$$N_c = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{pmatrix} -1 & 1 \\ -1 & 1 \\ 1 & -1 \end{pmatrix} \end{array}$$

There exists a weighting vector $W_c = (1, 1)^T$ such that $N_c \cdot W_c = 0$. Therefore, the control-loop net is strictly conservative. Since it is conservative, it is bounded too.

The reachability tree of the control-loop net is presented in Fig. 7. The boundedness property can be observed easily. The liveness and reachability properties can be determined by observing the occurrence of the marking of the frontier node at the end of each subtree, and the firing frequency of every transition. It is shown that the control-loop net satisfies the necessary properties.

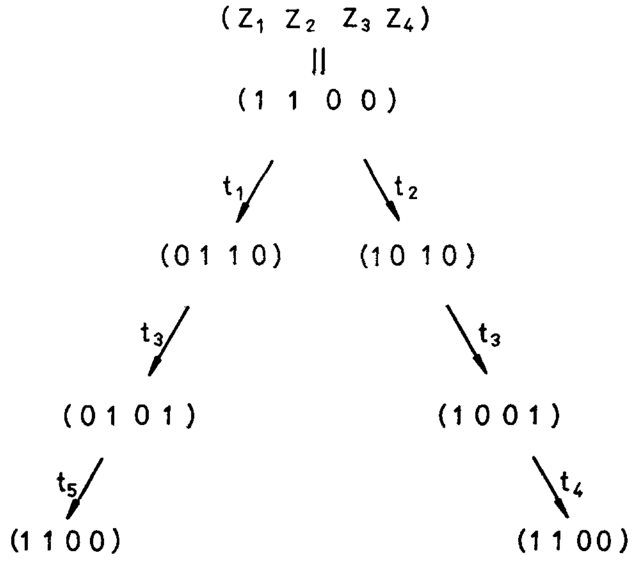

Fig. 6. The reachability tree of the modified flow-path net of the uni-directional merge structure.

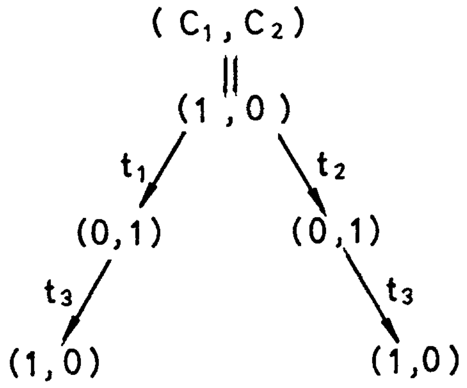

Fig 7. The reachability tree of the control-loop net of the uni-directional merge structure.

5.1.4. *Complete net analysis*

The flow-path net and the control-loop net are proven to be robust. The remaining issue which needs to be addressed is the condition of robustness for a complete net. Figure 4 shows the complete Petri net of a merge structure. Since it is not a closed loop, a modified net is needed. The modified complete net is not presented here since it can be obtained easily by combining the modified flow-path net in Fig. 5, and the control-loop net in Fig. 3. Because the complete net is composed of the two nets, the incident matrix can be estimated by combining the incident matrices of the two nets. The incident matrix can be obtained as

$$N = \left[N_f \;\middle|\; \frac{N_c}{\mathbf{0}} \right]$$

Let $W = [W_f | W_c]^T = (1, 1, 1, 1, 1, 1)^T$. Since $N_f \cdot W_f = 0$ and $N_c \cdot W_c = 0$,

$$N \cdot W = \left[N_f \cdot W_f \,\middle|\, \frac{N_c \cdot W_c}{\mathbf{0}} \right] = 0$$

There exists a weighting vector $W = (1, 1, 1, 1, 1, 1)^T$ such that $N \cdot W = 0$. Therefore, the merge net is strictly conservative.

The weighting vector W can be estimated directly from the incident matrix. To verify the above result, W is re-estimated. The incident matrix, N, of the merge net is given as follows:

$$N = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{c} \begin{array}{cccccc} Z_1 & Z_2 & Z_3 & Z_4 & C_1 & C_2 \end{array} \\ \begin{pmatrix} -1 & 0 & 1 & 0 & -1 & 1 \\ 0 & -1 & 1 & 0 & -1 & 1 \\ 0 & 0 & -1 & 1 & 1 & -1 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \end{pmatrix} \end{array}$$

There exists a weighting vector $W = (1, 1, 1, 1, 1, 1)^T$ such that $N \cdot W = 0$. The strictly conservative property of the net is again proven.

By studying the reachability trees (Figs 6 and 7), it is found that the transition firing sequences of the two trees are the same except for the last transition t_4 or t_5 in Fig. 6 which does not exist in Fig. 7. This is because two nets in fact share one set of transitions, and t_4 and t_5 can be enabled without any control tokens. Hence, the reachability tree of the complete merge net can be obtained easily by putting the markings (C_1, C_2) of the control-loop net after the markings (Z_1, Z_2, Z_3, Z_4) of the flow-path net to get the new markings $(Z_1, Z_2, Z_3, Z_4, C_1, C_2)$. For the last one in the reachability tree, add two zeros after the (Z_1, Z_2, Z_3, Z_4) marking. The safeness and boundedness properties of the complete net are of course there since the reachability tree is formed by combining Figs 6 and 7. Similarly, the reachability and liveness properties still exist after combination. For the purpose of verifying the above conclusions, the reachability tree of the modified complete net is drawn and presented in Fig. 8, where $t_{4\to1}$ represents the transition from Z_4 to Z_1, and $t_{4\to2}$ represents the transition from Z_4 to Z_2. The results support the conclusions made above.

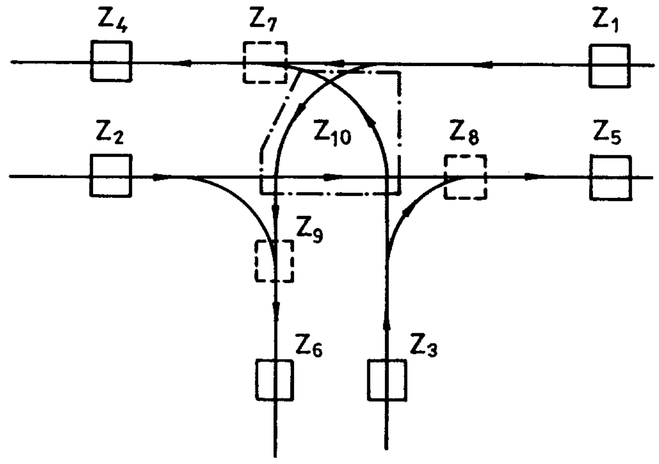

Fig. 9. The bi-directional merge flow-path layout.

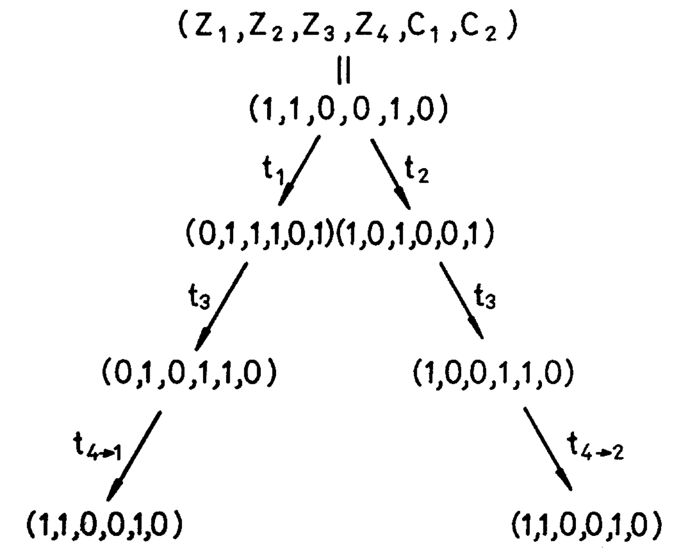

Fig. 8. The extended reachability tree of the uni-directional merge net.

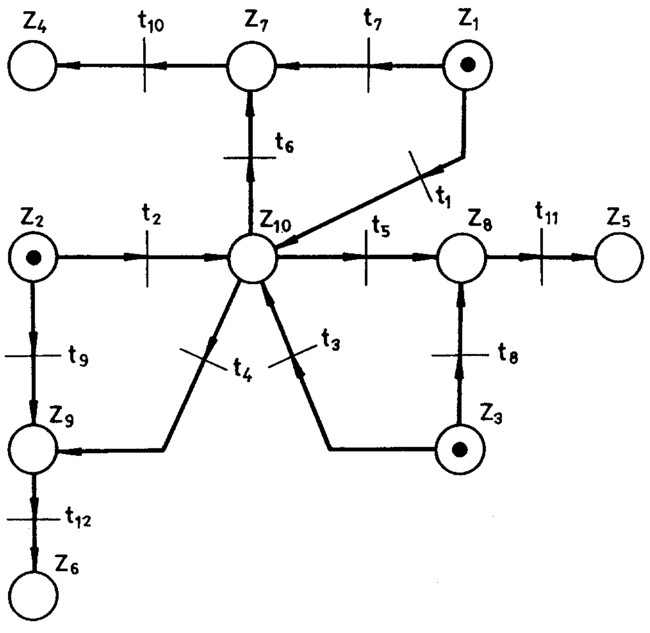

Fig. 10. The flow-path net of the bi-directional merge structure.

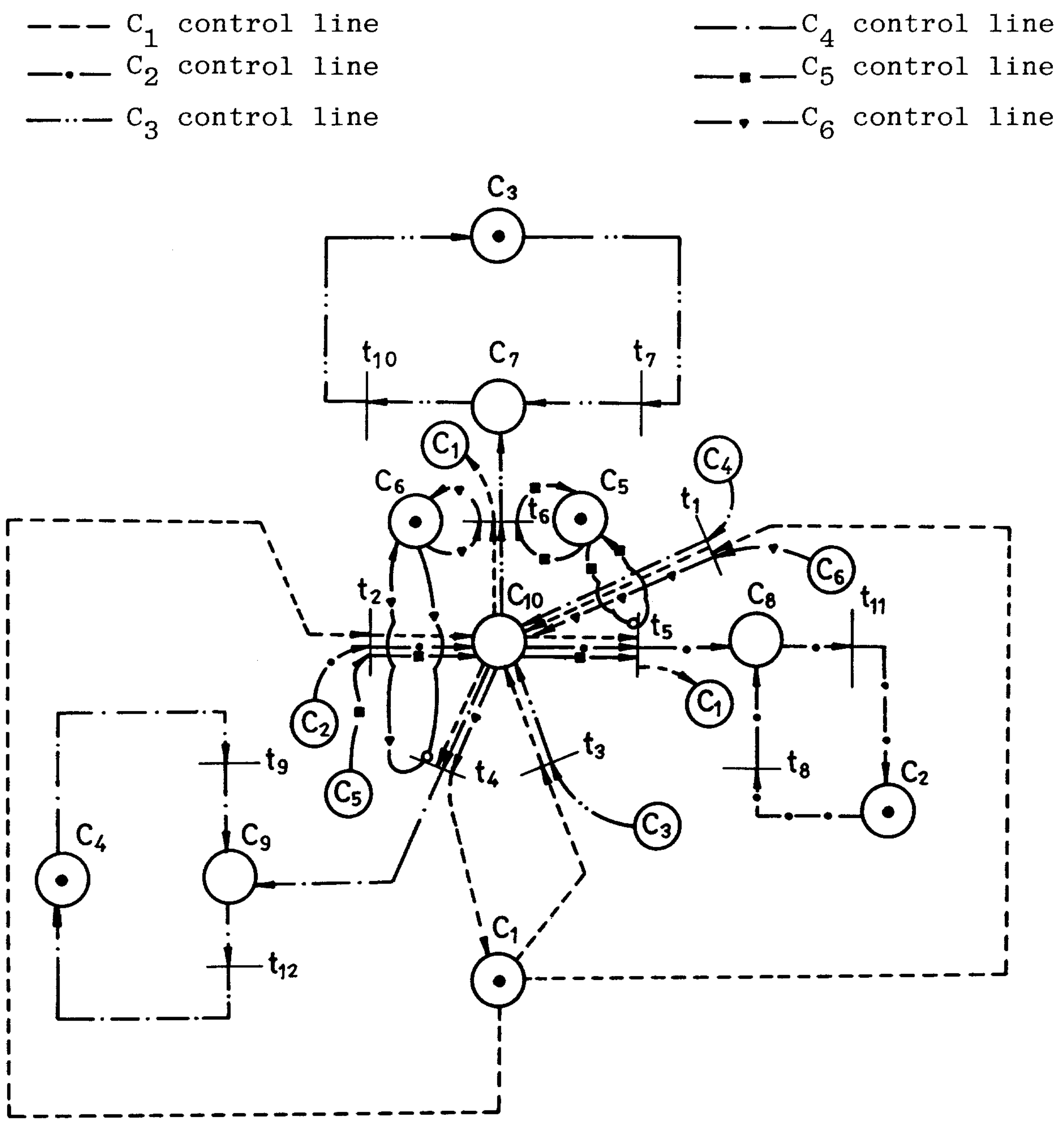

Fig. 11. The control-loop net of the bi-directional merge structure.

5.2. *Bi-directional merge case study*

A bi-directional merge case is used here as the second application example. The net construction and analysis procedures of the bi-directional merge case are similar to those of the uni-directional merge, so we will present the results only and leave out the details.

Figure 9 shows the flow-path layout of a bi-directional merge. According to the flow-path layout, the flow-path net is thus constructed and presented in Fig. 10. By observing the layout, three possible traffic problems may occur in this structure. They are:

(1) Z_1 vehicle wants to move to Z_6 while Z_2 vehicle wants to move to Z_5 or Z_6;

(2) Z_1 vehicle wants to move to Z_4 or Z_6 while Z_3 vehicle wants to move to Z_4;

(3) Z_2 vehicle wants to move to Z_5 while Z_3 vehicle wants to move to Z_4 or Z_5.

In order to avoid these three problems, by the use of logic circuit design concept (Mano, 1979), 10 more logic inputs need to be added. The control-loop net of the bi-directional merge structure is built and presented in Fig. 11. By placing the control-loop net over the flow-

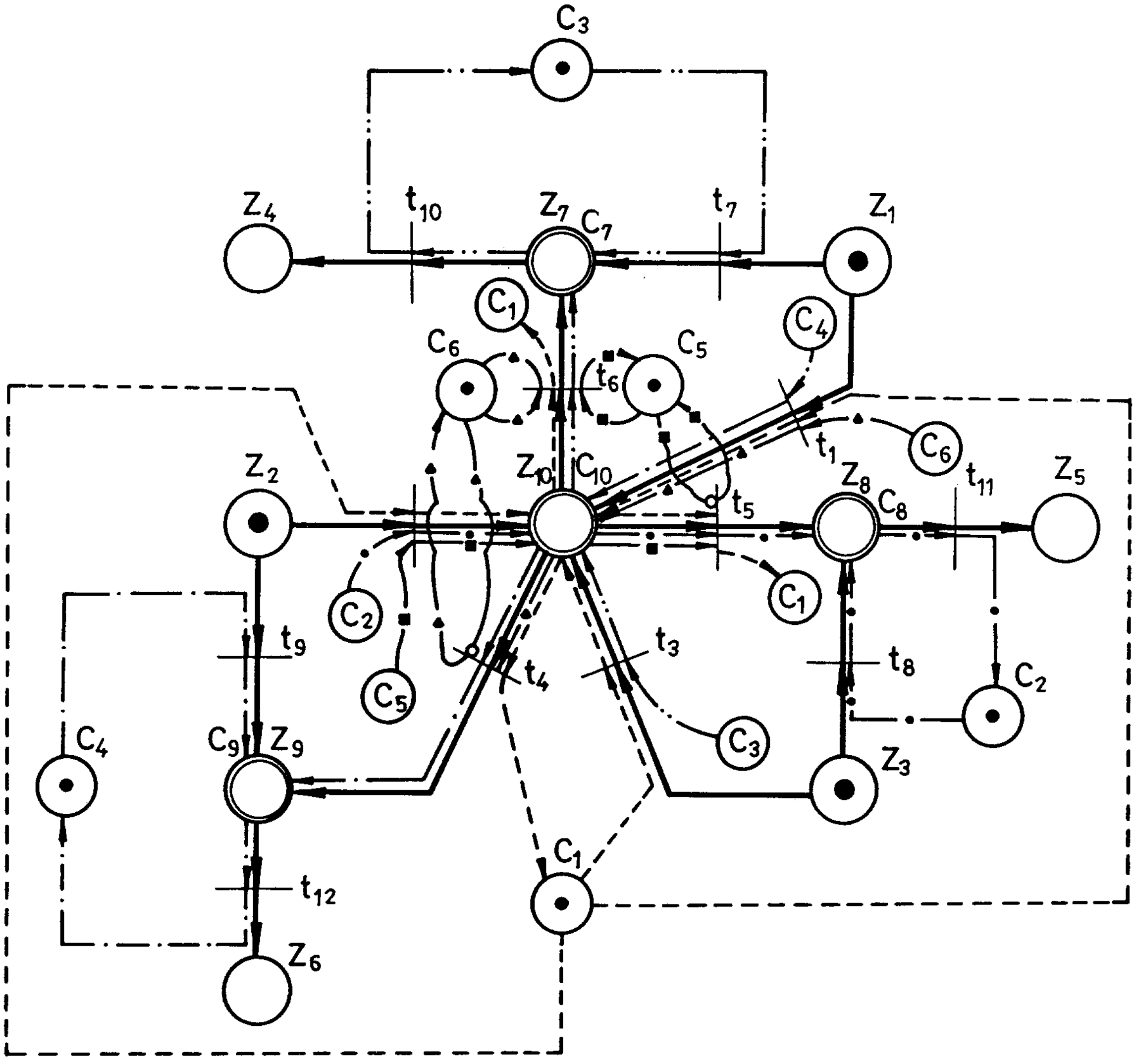

Fig. 12. The bi-directional merge net.

path net, the complete net is presented in Fig. 12. A quick method is used to validate the complete net, which is to synthesize the dynamic behaviour of the net by token plays. The result shows that all the three traffic problems mentioned are solved. This confirms the robustness of the net.

The following conclusions can be drawn from the study of the two examples presented above:

(1) The flow-path net is not a closed-loop system net, while the control-loop net is;

(2) The necessary properties can be embedded into the nets by adding extra places and tokens at proper locations;

(3) If those properties exist in both the flow-path net and the control-loop net, they also exist in the complete net.

6. Conclusions

This paper studied the fundamental system properties of a Petri-net based AGVS modelling system. Several necessary properties for building a robust AGVS at the floor level are characterized and presented. These properties include traffic collision-free, constant numbers of vehicles and traffic control signals, path reachability, and system deadlock-free. The Petri-net properties, including safeness, boundedness, strict conservation, reachability and liveness, are addressed in the light of constructing a better AGVS. Two different merge structures are presented to demonstrate how the properties in the net can be built and identified into an AGVS model. The results of this study provide the designer with a useful tool to construct a workable environment for building robust AGVS models.

References

Agerwala, T. (1979) Putting Petri nets to work. *Computer*, IEEE Computer Society, Los Alamitors, CA, December, 85–95.

AutoMod (1989) Auto Simulation Inc., Bountiful, Utah, USA.

Crockett, D., Desrochers, A., DiCesare, F. and Ward, T. (1987) Implementation of a Petri net controller for a machining workstation, report no. CH2413-3/87/0000/1861, IEEE.

Hsieh, S. and Lin, K.-H. M. (1991) AGVS mechanism, in *Proceedings of the 8th Conference of the Chinese Society of Mechanical Engineers in Taipei*, November.

Hsieh, S. and Lin, K.-H. M. (1992) Building AGV traffic-control models with place-transition nets, to appear in the *International Journal of Advanced Manufacturing Technology*.

Jensen, K. (1987) How to find invariants for coloured Petri nets, in *Lecture Notes in Computer Science*, Brauer, W., Reisig, W. and Rozenberg, G. (eds), Springer-Verlag, Germany, **118**, pp. 327–338.

Kamath, M. and Viswanadham, N. (1986) Applications of Petri net based models in the modelling and analysis of flexible manufacturing systems, report no. CH2282-2/86/000/0312, IEEE.

Mano, M. M. (1979) *Digital Logic and Computer Design*, Prentice Hall Inc., Englewood Cliffs, New Jersey, USA.

Martinez, J., Alla, H. and Silva, M. (1986) Petri nets for the specification of FMSs, in *Modelling and Design of Flexible Manufacturing Systems*, Elsevier Science Publishers B.V., Amsterdam, pp. 389–406.

Miner, R. J. and Rolston, L. J. (1987) *MAP/1 User's Manual*, Pritsker & Associates, Inc., P.O. Box 2413, West Lafayette, IN 47906, USA.

Peterson, J. L. (1981) *Petri Net Theory and the Modeling of Systems*, Prentice Hall Inc., Englewood Cliffs, N.J.

AGVs in America: an inside look (1990), in *Modern Material Handling*, Cahners Publishing Company, pp. 56–59.