Robustness Analysis of a Class of Non-ordinary Controlled Petri Nets
Fu-Shiung Hsieh, Member, IEEE

Abstract—Resource failure is a common problem in real systems. In case of resource failures, reanalysis of the perturbed system is required. Robustness analysis provides an alternative way to analyze a perturbed system without re-analysis. Robustness analysis aims to provide a quick and effective way to analyze a perturbed system without re-analysis.

I. INTRODUCTION

Many supervisory control problems in discrete event systems can be modeled and analyzed by Petri nets (CPN) [1]-[4]. In many problems, a resource is modeled by a token in Petri net model. Conservation of resource tokens makes it possible to derive many useful properties of Petri nets. Resource failure, which results from a variety of causes, including unreliable components, defective parts, faulty sensors, and so forth, is a common problem in real systems. Resource failures pose challenges in control of discrete event systems. In case of resource failures, reanalysis of the perturbed system is usually required. However, reanalysis based on existing reachability tree method is inefficient due to the state space explosion problem. Robustness analysis provides an alternative way to determine whether the operation of a perturbed system or part of it can still be maintained in case of resource failures. In this paper, we focus on robustness analysis of flexible assembly systems.

Assembly systems have attracted a lot of attention recently. For example, Fanti et al. [9] dealt with the deadlock problem in automated assembly systems based on discrete event system theory. Roszkowska and R. Wojcik [10] and Roszkowska [11] defined a Petri net model for assembly processes, where a control policy was proposed to determine the reserved buffer spaces to guarantee enough spaces for assembly to continue. Roszkowska also extended the results to compound processes in [12]. Despite the above researches, few address the robustness of assembly systems. In our recent work [5]-[8], we analyzed the robustness property of several subclasses of Petri nets, including controlled production Petri net (CPPN) [5], controlled assembly Petri net (CAPN)[6], controlled assembly/disassembly processes Petri nets (CADPN) [7] and controlled assembly Petri net with alternative routes (CAPN-AR) [8], where CPPN is a model for sequential processes, CAPN is for assembly processes and CADPN is a class for modeling systems with assembly/disassembly processes and CAPN-AR is a Petri net model for assembly processes with flexible routing. The aforementioned models and analysis of assembly systems are based on ordinary Petri nets. Extension of robustness analysis to non-ordinary Petri net models requires further study.

Non-ordinary Petri nets [13]-[14], which have weighted arcs, have the advantage to compactly model operations requiring multiple parts or resources. However, robustness analysis of non-ordinary Petri net model for flexible assembly processes has not been studied in existing literature. In this paper, we propose a non-ordinary controlled Petri with alternative routes (NCAPN-AR) model by extending the existing CAPN-AR model. NCAPN-AR exhibits properties that are different from CAPN-AR. As the weight of an arc in NCAPN-AR may be greater than one, firing a transition may require more than one token for each input place. Firing a transition may deposit more than one token into the output place of the transition. The minimal resource required to fire a transition depends on the arc weights. We characterize the robustness property for NCAPN-AR based on the condition of persistent production by exploiting the net structure.

The remainder of this paper is organized as follows. In Section 2, we introduce an uncertainties model in Petri nets. The NCAPN-AR model for flexible assembly processes and the condition for persistent production are presented in Section 3. In Section 4, a sufficient condition to test persistent production property and nominal supervisory control algorithm is proposed. We analyze the robustness of CAPN-AR in Section 5. We conclude this paper in Section 6.

II. NON-ORDINARY CONTROLLED PETRI NETS WITH UNCERTAINTIES

A non-ordinary controlled Petri net is defined as follows.

Definition 2.1: A non-ordinary controlled Petri net (NCPN) is defined as a seven-tuple \( G_{t} = (P,T,I,O,W,m_{0},u) \), abbreviated as \( G_{t}(m_{0},u) \), where \( T = T_{c} \cup T_{u} \), \( T_{c} \) is the set of controlled transitions, \( T_{u} \) is the set of uncontrollable transitions and \( u \) is a control policy defined based on control action of a given NCPN as follows.

Fu-Shiung Hsieh is with the Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taiwan, R.O.C. (e-mail: fshsieh@cyut.edu.tw).

This paper is supported in part by NSC under Grant NSC95-2416-H-324-009-MY2.
Definition 2.2: A control action \( a \) is a vector in \( Z^{|I|} \) that determines how many times that each transition in \( T_c \) may be fired concurrently, where \( a(t) \) denotes the number of times transition \( t \) can be fired under control action \( a \). A control policy \( u \) is a mapping \( u : R_c(m_0) \rightarrow Z^{|I|} \) that generates a sequence \( \{ a_n \} \) of control actions for the NCPN \( G_c \), based on its marking \( m \in R_c(m_0) \), where \( R_c(m_0) \) is the set of all reachable markings of \( G_c \) from an initial marking \( m_0 \).

Definition 2.3: \( G_c(m_0,u) \) is live if all the transitions in \( G_c \) can be fired infinitely from \( m_0 \) under the given \( u \). A control action \( a \) is called an admissible control action if there exists a control policy that keeps the NCPN live after executing \( a \).

To analyze the effects of uncertainties on a NCPN, we extended the definition of NCPN as follows.

Definition 2.4: A NCPN with uncertainties (NCPNU) \( G_c^{\Delta}(m_0,u) = (P,J,I,O,W,R_c,\Delta,\Delta_c,\mu_G) \) is an eight-tuple, where \( P \) is a finite set of places with cardinality \( |P| \), \( T \) is a finite set of transitions, \( \Delta \subset P \times T \) is a set of transition input arcs, \( O \subset T \times P \) is a set of transition output arcs, \( W : I \cup O \rightarrow \{1,2,3,...\} \) is a weight function, \( \Delta : R_c(m_0) \rightarrow Z^{|I|} \) is a mapping that specifies the maximal perturbation \( \Delta(m) \) for each marking \( m \in R_c(m_0) \), with \( 0 \leq \Delta(m) \leq m \) and \( \Delta(m) \in Z^{|I|} \), \( m_0 : P \rightarrow Z^{|I|} \) is the initial marking of the PN with \( Z \) as the set of nonnegative integers, and \( u \) is a mapping \( u : R_c(m_0) \rightarrow Z^{|I|} \) that generates a sequence \( \{ m' \} \) of control actions for CPNU \( G_c^{\Delta} \) from \( m_0 \), where \( R_c^{\Delta}(m_0) = \{ m' \mid m' \in R_c(m - \delta) \} \), where \( m \in R_c(m_0) \), \( 0 \leq \delta \leq \Delta(m) \), \( 0 \leq \Delta(m) \leq m \) is the set of all reachable markings of \( G_c^{\Delta} \) from \( m_0 \).

Liveness of a NCPNU \( G_c^{\Delta} \) is defined as follows.

Definition 2.5: \( G_c^{\Delta} \) is live under marking \( m_0 \) if there exists a control policy \( u' \) under which \( G_c^{\Delta}(m_0,u') \) is live for each \( m' \in R_c^{\Delta}(m_0) \).

Definition 2.6: For two uncertainty functions \( \Delta_1 \) and \( \Delta_2 \), we say \( \Delta_1 \) dominates \( \Delta_2 \), denoted by \( \Delta_1 \preceq \Delta_2 \), if \( \Delta_1(m) \geq \Delta_2(m) \) for each \( m \in R_c(m_0) \). We say \( \Delta_1 \) strictly dominates \( \Delta_2 \), denoted by \( \Delta_1 \rightarrow \Delta_2 \), if \( \Delta_1 \geq \Delta_2 \) and \( \Delta_1(m) > \Delta_2(m) \) for some \( m \in R_c(m_0) \).

Definition 2.7: An uncertainty function \( \Delta_1 \) is a maximal tolerable uncertainty function if \( G_c^{\Delta_1} \) is live under marking \( m_0 \) and \( G_c^{\Delta_2} \) is not live under marking \( m_0 \) for any uncertainty function \( \Delta_2 \) with \( \Delta_2 \geq \Delta_1 \).

If \( G_c^{\Delta} \) is live under marking \( m_0 \), \( G_c^{\Delta'} \) must be live under marking \( m_0 \) for any \( \Delta' \leq \Delta \).

In a system with alternative routes, it is not required for the NCPN \( G_c^{\Delta} \) to be live to maintain the operation. To characterize the condition that maintains the operation of a system without enforcing the liveness of the entire net, the concept of persistent production is introduced. The concept of persistent production was originally proposed in [8]. Let \( T_f \subseteq T \) denote the set of final transitions that represent the completion of a task in a NCPN. In this paper, we extend the concept of persistent production for a NCPN as follows.

Definition 2.8: A NCAPN-AR \( G_c^{\Delta} \) is persistent under marking \( m_0 \) if and only if there exists a control policy under which each transition in \( T_f \subseteq T \) is live from \( m_0 \).

Definition 2.9: \( G_c^{\Delta} \) is persistent under marking \( m_0 \) if there exists a control policy \( u \) under which \( G_c^{\Delta}(m',u) \) is persistent for each \( m' \in R_c^{\Delta}(m_0) \).

Given a NCPN, an important issue is to find the maximal tolerable uncertainty function \( \Delta \). However, the computational complexity to find the maximal tolerable uncertainty function of a general NCPNU grows exponentially with the size of the nets. Instead of considering a general NCPNU, we limit the scope and concentrate on the robustness of a class of Petri nets called NCAPN-AR (Non-ordinary Controlled Assembly Petri Net with Alternative Routes).

III. NCAPN-AR MODEL

Construction of a NCAPN-AR is done by a bottom-up approach. First, the Petri net model of the process workflow and the Petri net model of the resources are constructed separately. To capture the interactions among resources and process workflows, we apply the operation “\( \parallel \) ” (defined in [9]–[12]) to merge two PNs through common places, transitions, or arcs. The Petri net model \( G_c = \bigsqcup_{c \in C} G_{c,c} \parallel_{c \in J} G_{c,c} \) is constructed by merging the resource subnets \( G_{c,r} \), \( r \in R \), with job subnets \( G_{J,j} \), \( j \in J \), defined as follows, where \( J \) denotes the set of different types of processes in the system and \( R \) denotes the set of resource types in the system, defined as follows.

Definition 3.1: A type- \( j \) job subnet \( G_{J,j} = (P^j_I, T^j_I, I^j_I, O^j_I, W^j_I, m_{j,0}) \) is an acyclic Petri net of tree structure, where \( m_{j,0}(p) = 0 \) \( \forall p \in P^j_I \). That is, no job is in process under \( m_{j,0} \). We use \( t^j_f \) to denote the unique final transition. Each place in \( P^j_I \) has exactly one output transition but may have multiple input transitions. We use \( \hat{P}^j \) to denote the subset of places in \( P^j_I \) with multiple input transitions. We define \( T_f^j = \{ t^j_f | j \in J \} \).
A type- \( r \) resource may take part in a set \( \Omega_r = \{1, 2, 3, \ldots, K_r\} \) of activities. To model an activity \( k \in \Omega_r \) in Petri net, we use a transition \( t_{dk}^k \) to represent allocation of resources and use \( t_{dk}^k \) to represent de-allocation of resources after completing the operations in it. The \( k \)-th activity of type- \( r \) resources is described by a directed path \( p_1 t_1 p_2 t_2 p_3 \ldots p_d t_d p_{d+1} \) starting with place \( p_{d}^k \) and ending with place \( p_{d+1}^k \), where \( p_1 = p_{d}^k \), the idle state before allocating a type- \( r \) resource, \( p_{d+1} = p_{d}^k \), the idle state after de-allocating a type- \( r \) resource, \( t_i = t_d^k \) and \( t_n = t_d^k \) .

Let \( w_i = W(p_i, t_i) \) and \( v_i = W(t_i, p_{i+1}) \) for \( i = 1, 2, 3, \ldots, n \) . We assume \( \frac{v_{n-1} v_{n-2} \ldots v_{n-k}}{w_n w_{n-1} \ldots w_{n-k+1}} \frac{1}{w_1} = 1 \). The Petri net model for the \( k \)-th activity is described by a Petri net \( G_r^k \) defined as follows.

Definition 3.2: The \( k \)-th activity of type- \( r \) resources is represented by the Petri net \( G_r^k = (P_r, T_r, I_r, O_r, W_r, m_{r_0}^k) \). Remark that \( T_r \cap T_r = \Phi \) for \( k \neq k' \).

For each \( r \in R \), we merge places \( p_{d}^k \) and \( p_{d}^k \) for all \( k \in \Omega_r \) into one place \( p_{d}^k \). The type- \( r \) resource subnet \( G_r \) is defined by \( G_r = \bigcup_{k \in \Omega_r} G_r^k = G_r^1 \| G_r^2 \| \ldots \| G_r^k_r \) as follows.

Definition 3.3: The Petri net \( G_r = \bigcup_{k \in \Omega_r} G_r^k = (P_r, T_r, I_r, O_r, W_r, m_{r_0}^k) \) denotes the type- \( r \) resource subnet, where \( m_{r_0}^k(p_{d}^k) > 0 \) and \( m_{r_0}^k(p) = 0 \) \( \forall p \in P_r \setminus \{p_{d}^k\} \) .

\( P_o = \{p_{d}^k \} \) denotes the set of all resource idle places.

Figure 1(a) illustrates a NCAPN-AR \( G_e \), where the number in each place denotes the number of tokens. \( G_e \) may evolve to a new marking as shown in Figure 1(b).

Definition 3.5: Let \( j \in J \). A NCAPN-AR \( G_i \) is \( j \)-persistent under a marking if and only if there exists a control policy under which transition \( t_j^i \) is live. A NCAPN-AR \( G_i \) is persistent under a marking if and only if \( G_i \) is \( j \)-persistent for all \( j \in J \).

Definition 3.6: \( G_i^k \) is \( j \)-persistent under marking \( m_0 \) if there exists a control policy \( u \) under which \( G_i(m^i, u) \) is \( j \)-persistent for each \( m^i \in R_i^k(m_0) \).

Definition 3.7: \( M^j \) denotes the set of initial markings of \( G_i \) with minimal resources for the existence of a control policy to keep \( t_j^i \) live for all \( j \in J \), where \( M_j^i \subset M_0 \). The set of resources under \( m^j \in M_j^i \) is called a minimal resource requirement (MRR) of type- \( j \) jobs.

In this paper, we assume \( m_0 \geq m = m_j \oplus m_j \oplus m_j \oplus \ldots \oplus m_j \). Property 3.1: Given a NCAPN-AR \( G_e \) with marking \( m \in R(m_0) \), there exists a control policy \( u \) such that \( G_e \) is persistent under \( m \) if and only if there exists a sequence of control actions that bring \( m \) to a marking \( m^i \in M_0 \) with \( m^i((p_{d}^k)) \geq m^i((p_{d}^k)) \) \( \forall r \in R \), \( \forall j \in J \), where \( m^j \in M_j^i \).

Figure 1(b) A NCAPN-AR \( G_e \)

Application of Property 3.1 requires computation of \( m^j \) and testing reachability of a marking \( m' \) that covers \( m^j \) \( \forall j \in J \). A heuristic to obtain a lower bound of \( m^j \) can be found by firing an arbitrary permutation of the transitions in \( T_j^i \). Direct application of reachability tree method to this problem is computationally feasible only for small Petri nets [1]-[2]. Instead, we propose an efficient but sufficient only test procedure by exploiting the net structure.
IV. TESTING PERSISTENT PRODUCTION PROPERTY

We test whether \( m^* \) is coverable by calculating the set of resources that can be returned to idle state by firing as many transitions as possible under a marking \( m \) for each \( j \in \mathcal{J} \).

Property 4.1: \( GJ \) can be decomposed into a set \( D_j \) of subprocesses by splitting each place \( p \in \hat{P}_j \) into \( \{ p \cup p^* \} \) places. Let \( GJ_d = (P^d, T^d, I^d, O^d, m^d) \) be the Petri net of subprocess \( d \in D_j \).

Based on \( D_j \), we construct the Petri net \( MG_d = \bigsqcup \mathcal{G}_k \bigsqcup GJ_d \) by merging the Petri nets \( G^J \) with the Petri net \( GJ_d \) to model the interactions between the resource and type- \( j \) jobs. The marking of \( MG_d \) is denoted by \( m_d \) and is obtained by projecting marking \( m \) onto the places of \( MG_d \) according to the control action \( a \).

The \( MG_d \) corresponding to Figure 1(b) is shown in Figure 2.

Definition 4.1: \( MG_d = \bigsqcup \mathcal{G}_k \bigsqcup GJ_d \) for each \( j \andleft( P_d, T_d, I_d, O_d, m_d \right) = \left( P^d \cup P^d \cup P^d, T^d \cup T^d, I^d \cup I^d, O^d \cup O^d, m_d \right) \), where \( P^d = \bigcup \mathcal{G}_k \) and \( \mathcal{G}_k \) denotes the set of idle state places to allocate a type- \( k \) resource. \( \mathcal{G}_k \) denotes the set of activities in subprocess \( d \) and \( P^d_r = \bigcup \mathcal{G}_k \) and \( P^d_r = \{ p^d_k \mid k \in \mathcal{G}_k \} \) denotes the set of idle state places after de-allocating a type- \( r \) resource, respectively.

The Petri net \( MG_d \) in Figure 2. \( P^d_d \{ p^d_1 \} \) and \( P^d_d \{ p^d_2 \} \) is a directed path of \( MG_d \), where \( p^d_i \in P^d_r \) and \( p^d_i \in P^d_r \). \( \Pi (p) \) denotes the set of token flow paths in \( MG_d \) ending with place \( p \).

In Fig. 2, \( \Pi \{ p^d_i \} \{ \pi_1, \pi_2 \} \), where \( \pi_1 = (p^d_1) \) and \( \pi_2 = (p^d_2) \), \( \Pi \{ p^d_i \} = \{ \pi_3, \pi_4, \pi_5 \} \), where \( \pi_3 = (p^d_1) \), \( \pi_4 = (p^d_2) \), \( \pi_5 = (p^d_1) \). To compute token flow, we define the following notations.

Definition 4.3: We use \( [x] \) to denote the maximal integer no greater than \( x \). \( \lceil x \rceil \) if \( n \leq x < n+1 \).

Definition 4.4: Suppose \( \alpha = \alpha(p^d_1, p^d_2, p^d_3, \ldots, p^d_n) \). Let \( w_i = W(p_i, t_i) \) and \( v_i = W(t_i, p_{i+1}) \) for \( i = 1, 2, 3, \ldots, n \). We define

\[
\alpha_{p_{n-1}}(m_d) = v_{n-1} \frac{v_{n-2}}{w_{n-1}} \ldots \frac{v_{n-k}}{w_{n-k+1}} \frac{m_d(p_{n-k})}{w_{n-k}}]
\]

where \( k = 0, 1, 2, \ldots, n-1 \).

We define \( \pi(m_d) = m_d(p) + \sum_{k \in \{1, 2, 3, \ldots, n\}} \alpha_{p_{n-1}}(m_d) \).

The following lemma holds.

Lemma 4.1: Given \( MG_d \) under marking \( m_d \), the maximum number of tokens that will flow to place \( p \in P^d_r \) is no less than \( \gamma_p(m_d) = \min_{\pi \in \Pi(p)} \pi(m_d) \), where \( r \in R \).

\[
\pi_1(m_d) = m_d(p^d_1) + \alpha_{p^d_1}(P^d_1) + \alpha_{P^d_1}(P^d_1) = m_d(p^d_1) + \frac{m_d(p^d_1)}{2} + \frac{m_d(p^d_1)}{2} = 8
\]

\[
\pi_2(m_d) = m_d(p^d_1) + \alpha_{P^d_1}(P^d_1) = m_d(p^d_1) + \frac{m_d(p^d_1)}{3} + \frac{m_d(p^d_1)}{3} = 2.
\]

\[
\gamma_p(m_d) = \min_{\pi \in \Pi(p)} \pi(m_d) = \min(\pi_1(m_d), \pi_2(m_d)) = 2
\]

Property 4.1: If \( MG_d \) is an ordinary Petri net, \( \pi(m_d) = \sum_{p \in \pi} m_d(p) \), where \( p \in \pi \) means that \( p \) is a place in \( \pi \). That is, \( \pi(m_d) \) is equal to the total number of tokens in \( \pi \). In this case, \( \gamma_p(m_d) = \min_{\pi \in \Pi(p)} \sum_{p \in \pi} m_d(p) \).
The number of type- \( r \) resources that will be released to the set \( P^o_d \) of places after by firing all the transitions in \( T_d \) as many times as possible under \( m_d \) is \( \sum_{p \in P^o_d} \gamma_r(p) \).

By summing up \( \sum_{p \in P^o_d} \gamma_r(p) \) for each \( d \in D \), the number of type- \( r \) resources that will be released to the set \( P^o_d \) of places after firing all the transitions in \( T_d \) as many times as possible under marking \( m_d \) is \( \sum_{j=0}^{n} \sum_{d \in D} \sum_{p \in P^o_d} \gamma_r(p)(m_d) \).

For each \( \delta \), and \( J \) denote the \( \delta \), where \( \delta \) and \( \delta \). A safe margin for making \( \delta \) in nominal marking \( J \) and \( J \).

If \( J \) otherwise. If \( J \) can be \( \gamma \) and \( J \)

We are \( \pi \) \( J \) . We are \( \pi \) \( J \).

V. ROBUSTNESS ANALYSIS

We model resource unavailability as a perturbation, \( \delta \), \( 0 \leq \delta \leq \Delta(m) \) in nominal marking \( m \). Depending on the structure of \( \delta \), its impacts on the execution of a task vary. In this section, we analyze the effects of \( \delta \) by exploiting the structure of \( MG_d \). We establish the conditions under which Property 4.2 is conserved. We first consider perturbation \( \delta \) in a single place \( p \). In this case, \( \delta \in \Delta(m) \), \( \delta(p) > 0 \) and \( \delta(p') = 0 \) otherwise. The perturbation \( \delta \) can be broken down into components \( \delta_d \), \( d \in D_d \), \( j \in J \). In this case, \( \delta_d = 0 \) \( \forall d' \neq d \), \( \delta_d(p) > 0 \) and \( \delta_d(p') = 0 \) otherwise. Let \( d \in D_d \) be the subprocess to which place \( p \) belongs.

Let \( p' \in P^o_d \) and \( \gamma_{p'}(m_d) = \min_{\pi} \pi(m_d) \). We are interested in the condition under which \( \gamma_{p'}(m_d - \delta_d) = \gamma_{p'}(m_d) \). A safe margin for making this condition valid is stated as follows. Let \( N_p = \{ p' | p'^* = p \} \) and let \( w_p = W(p, t) \), where \( r \) denotes the output transition of \( p \).

Property 5.1: Suppose \( \delta_d = 0 \) \( \forall d' \neq d \), \( \delta_d(p) > 0 \) and \( \delta_d(p') = 0 \) otherwise. If \( 0 \leq \delta_d(p) \leq \gamma_{p'}(m_d) - w_p c^*_p(m_d) \), \( \gamma_{p'}(m_d - \delta_d) = \gamma_{p'}(m_d) \), where \( p' \in P^o_d \), \( \gamma_{p'}(m_d) = \min_{\pi} \pi(m_d) \) and \( c^*_p(m_d) = \min_{p \in P^o_d} [\gamma_{p'}(m_d)] \).

Let \( p \in P \) and \( \Delta_p(m) = \{ \delta | \delta \in \Delta(m), 0 < \delta \leq \min(m_d(p), \gamma_{p'}(m_d) - w_p c^*_p(m_d)) \}, \delta(p') = 0 \) \( \forall p' \in P \setminus \{ p \} \).

Based on Property 5.1, the following theorem holds.

Theorem 5.1: Suppose NCAPN-AR \( G \) is persistent under marking \( m_0 \), where \( m_0 \geq m_j \forall j \in J \). Let \( u_{nc} \) denote the
nominal supervisory control algorithm. Suppose $m \in R_e(m_0)$ is a nominal marking reached under $u_{ncs}$. $G^{\Delta \epsilon}_e$ is persistent under marking $m$.

Consider place $p_1$ in Figure 2, $N_{p_1} = \{p_{1j}, p_1\}$. 

$$c^*_p(m_{d_j}) = \min_{p' \in N_{p_1}} \left[ \frac{\gamma_{p_1}(m_{d_j})}{\gamma_{p_1}(m_{d_j})} \right] \frac{w_{p_{1j}}}{w_{p_1}}$$

$$= \min_{p' \in N_{p_1}} \left[ \frac{3}{4} \right] = 1$$

$$\min(m_{d_j}(p_1), \gamma_{p_1}(m_{d_j}) - w_{p_1}c^*_p(m_{d_j})) = 3$$

$$\Delta_p(m) = \{|\delta| \in \Delta(m), 0 < \delta(p_1) \leq 3, \delta(p') = 0 \forall p' \in P \setminus \{p_1\}\}$$

$G^{\Delta \epsilon}_e$ is persistent live under marking $m$.

Theorem 5.2: Suppose NCAPN-AR $G_e$ is $j$-persistent under marking $m_0$. Let $u_{ncs}$ denote the nominal supervisory control algorithm. Suppose $m \in R_e(m_0)$ is a nominal marking reached under $u_{ncs}$. $G^{\Delta \epsilon}_e$ is $j$-persistent under marking $m$.

By applying Property 5.1, we analyze perturbation in multiple arcs as follows. Let

$$\Delta_F(m) = \{|\delta| \in \Delta(m), 0 < \delta(p) \leq 3, \delta(p') = 0 \forall p' \in F \setminus \{p_1\}\}$$

Theorem 5.3: Suppose NCAPN-AR $G_e$ is persistent under marking $m_0$, where $m_0 \geq m^*_j \forall j \in J$. Let $u_{ncs}$ denote the nominal supervisory control algorithm. Suppose $m \in R_e(m_0)$ is a nominal marking reached under $u_{ncs}$. $G^{\Delta \epsilon}_e$ is persistent under marking $m$.

VI. CONCLUSION

Robustness analysis of Petri nets paves the way for dealing with uncertainties in discrete event systems. In our previous papers, we have unveiled robustness properties for ordinary controlled Petri nets. Non-ordinary Petri nets have weighted arcs and have the advantage to compactly model operations requiring multiple parts or resources. In this paper, we extend the robustness analysis method to a subclass of non-ordinary controlled Petri nets (NCPN) called non-ordinary controlled Petri with alternative routes (NCAPN-AR). Due to the existence of alternative routes in a NCAPN-AR model, it is not required for the net to be live to maintain production. The concept of persistent production is introduced to capture this feature of NCAPN-AR. We propose a nominal supervisory control algorithm to enforce persistent production for a NCAPN-AR in nominal state. We propose sufficient persistent production conditions for NCAPN-AR in the presence of uncertainties. A nominally persistent NCAPN-AR is still persistent under a perturbed state as long as the sufficient condition holds.

REFERENCES


