A Report on – “A Banker’s Solution for Deadlock Avoidance in FMS With Flexible Routing and Multiresource States” by J. Ezpeleta, F. Tricas, F. García-Vallés, and J. M. Colom

This paper defines a class of Petri nets called *S*PR* which can formally model Flexible Manufacturing Systems. These nets are natural extensions of previous classes of nets already researched upon. They model sequential Resource Allocation Systems with –

- Routing flexibility in the processing of parts
- Multiset of resources at each processing step of a part

In the models considered here, each state change implies a resource allocation operation where a set of resources are allocated and a set of resources are freed. The resources are **serially reusable** which means that the number of resources is constant and each resource unit is either available or allocated to a single process. A resource unit may be released by a process only if it has acquired it previously.

Class of *S*PR* Nets

Few properties about some components of the *S*PR* net model structure –

- It is a connected generalized self-loop free Petri Net $N=<P,T,C>$ where $P = P_0 \cup P_S \cup P_R$
  P_S denotes **process state places** and a token in a place models a part which is in-process, at a reachable marking
  P_0 represents states in which corresponding processes/parts are idle
  P_R denotes **resource places** and model how resources can be used by active processes
- The subnet generated by places P and transitions T is a strongly connected state machine.
- This enforces the existence of a minimal P-Semiflow Y_r such that $Y_r(r) = 1$.
- For a given process state place, $Y_r(p) = k$ which means that k copies of resource r are used by each process (token) modeled by place p.
- Resources can neither be created nor destroyed.
- Resources are serially reusable.
- The initial markings must represent system idle state. This means that the initial marking of process places must be 0, the initial marking of each idle place represents the maximal number of parts of the type modelled by the state machine that are allowed to be concurrently processed and that all resources are available initially. Thus, the initial markings of P_R will correspond to the resource capacities which is greater than 0.
- The system must be well defined i.e. each possible processing sequence for any given part can be executed in isolation.
- M_R denotes the multiset of free resources at a reachable marking M.
- A transition t is enabled at M if the input state place is marked i.e. the transition is **process enabled** and each input resource place also enables the transition i.e. it is **resource enabled**.

A Banker's Algorithm for Deadlock Avoidance in *S*PR* Nets

Apart from this, the paper also extends the classical Bankers algorithm for deadlock avoidance in *S*PR* nets. A deadlock basically means that the processing of parts cannot finish once started because each and every part requests for some resource that is currently held by some other part in this set. A **deadlock avoidance** approach is followed in this paper which means that in the granting of resources to any process, the controller must ensure that it will lead to **safe state** i.e. a state from which all parts currently being processed can terminate. To do this, the maximum need of resources throughout the life of an active process, the set of available resources and the resources assigned to each process needs to be known.

This algorithm decides whether a state is safe and then ensures that all the in-process parts can be terminated in a sequential manner. The worst-case complexity of this algorithm is that it is **polynomial** on the Petri net model size. The proposed algorithm is applicable for those sequential RAS where transitions related to the granting of resources can be controlled and the set of states a part can stay during its processing is known. The set of states can be modelled as a state machine so that any cycle which contains the initial state is a valid production sequence for a part. Also, all paths which connect the initial state and a given state are equivalent and valid alternatives to each other.

The basic outline of the algorithm is as follows. To verify whether a given system is safe –

1) We find an active process that is able to terminate using the resources it holds plus the available resources.
2) If no such process exists, the state is considered to be unsafe.
3) If such a process exists, add the resources used by this process to the free resources, withdraw the process form the set of active processes and iterate.
4) If every process can be removed from the set of active processes, the state is safe.

When applied to *S*PR* nets, there are two parts to this algorithm. The first part verifies whether there are enough resources for the process termination when the rest of the active processes do not move from their current state. It basically determines whether an active process corresponding to a reachable marking M is M_R-terminable. A process is ***M_R*-terminable** if there exists a path joining the state place where the process resides and its corresponding idle state, such that this path can be followed by the process using the resources that are free and those it is currently holding.

The second part of the algorithm checks whether a reachable marking is safe i.e. an ordering for the sequential termination of the active processes can be found. This part makes use of the fact that if at a given state a process can terminate, then all other processes in the same state place can terminate one after another. This is only a sufficient condition for the state to be safe. This part determines the overall complexity of the algorithm. The worst case corresponds to when every state place is marked at M. This results in the algorithm being polynomial in the Petri net model size.