REPORT ON ‘A BANKER’S SOLUTION FOR DEADLOCK AVOIDANCE IN FMS WITH FLEXIBLE ROUTING AND MULTIRESOURCE STATES’

Veena Tella 2016A7PS0847H

Mathangi Sundar 2016A7PS08180H

Rut Tapan 2016A7PS0052H

INTRODUCTION

Flexible Manufacturing System(FMS) are systems where there is space for some amount of flexibility that allows the system to react in case of changes, whether predicted or unpredicted. Petri nets are a formal model used in FMS. in non assembly systems, Petri net model is based on a set of state machines, each one modelling process plans. These state machines aren’t independent and each state needs resources to carry out the

operations modelled by a state. In the models considered here, each state change corresponds to an operation in which sets of resources are freed or engaged. These systems are called Resource Allocation Systems(RAS). There are different categories of RAS mainly differentiated on the basis of usage of resources and structure of set of processing sequences.

Deadlock is a situation where two processes prevent each other from using resources because these resources are held by them. There are different ways to prevent or avoid a deadlock. In a deadlock avoidance approach, we first check if the resulting state is a safe state before granting the resource. In the case of the Banker's Algorithm, we need to know about four data structures- Available, Maximum needed, Allocation and Need.

CLASS OF S*PR NETS

S*PR nets is a class of Petri nets, that can deal with sequential RAS with routing flexibility and use of multiple copies of different resources per state.

Definition 1: Let IN = {1, 2 ...m} be a finite set of indexes. An SPRnet is a connected generalized self-loop-free Petri net N = {P, T, C} where

1) P = P0 U PS U PR is a partition such that:

a) PS = U(for i belongs to In) PSi , PSi= not null, and PSi (intersection)PSi = not null, for all i !=j;

b)P0 = U(for i belongs to In) {po}

c) PR = {r1,r2,r3.....rn}, n>0

2) T = U(for i belongs to In)Ti, Ti= not null, Ti intersection Tj = not null , for all i != j

3) For all i belongs to In , the subnet Ni generated by PSi U {po} U Ti is a strongly connected state machine.

4) For each r belongs to Pr, there exists a unique minimal P-Semiflow, Yr belongs to IN^||P||, such that {r} = ||Yr||intersection Pr, Yr(r)=1 , P0 intersection ||Y|| = not null, and PS intersection ||Yr|| = not null.

5) PS = U(r belongs to Pr) ||Yr||\{r}.

Places of Ps are called process state places. Each place Po-i is called idle state place and it represents the state in which the corresponding processes are idle. Pr places are called

resource places, which represent how resources can be used by the active processes. According to the definition, it shows that resources can neither be created nor destroyed. Yr represents a minimal P-semiflow. These P-semiflows impose the resources to be serially reusable. The initial markings representing no activity in the system and allowing the processing of each part in isolation are accepted. These initial markings are made accordingly-

1) They represent the system idle state. This means that:

a) the initial marking of process places must be 0 (there is no activity in the initial state

b) the initial marking of each idle place represents the maximal number of parts of the type modeled by the state machine that are allowed to be concurrently processed

c) considering the previous points, all resources are available. Consequently, the initial markings of places in PR correspond to the resource capacities, and therefore, are greater than 0.

2) The system is well defined. This means that each possible processing sequence for any given part can be executed in isolation.

3.Must satisfy - Definition 2: Let {N, M0} be a S*PR. An initial marking M0 is acceptable for N if and only if

1) For all i belonging to In , M0(p0) > 0.

2) For all p belonging to Ps, M0(p)=0 .

3) For all r belonging to Pr, M0(r) >= max(p belongs to H) Yr(p).

A BANKER'S ALGORITHM FOR DEADLOCK AVOIDANCE IN S*PR NETS

In the algorithm, each active process has to be identified. As stated above, in a SPR model, a process is a token in a state place. Let us now denote by |M| the number of active processes at marking M: |M| = sigma(p belongs to Ps) M(p). Also, associated to each reachable marking, Am = {a1,a2….a|m|} will denote the set of tokens in state places (active processes), while **Π**m:Am -> Ps defines the mapping relating each active process to its corresponding state place.

In check if the given system state is safe, we have to search for an active process able to

terminate using the resources it holds plus the available resources. If no such process exists, the state is considered to be non-safe. If it exists ,add the resources used by such process to the free resources, withdraw the process from the set of active process, and iterate the method. If every process can be removed from the set of active processes, the state is safe. A process is said to be Mr-terminable if there exists a path joining the state place where the process stays and the corresponding idle state, in such a way that the path can be followed by the process using the free resources plus those it is currently holding.

The algorithm to check if an active process is Mr-terminable is-

```
Function isTerminable
    (In N, a marked S*PR;
    In M: a (RS)M, N_i, M_a{;
    In a: an active process)
Return (iT:Boolean)
- -  Pre: a ∈ M_A, π_R(a) p iP_i
- -| Post: iT= Is a M_R-Terminable?
Begin
    For Each p ∈ P_{S_i}
      If M_R + Y_R(π_M(a)) − Y_R(p) ≥ 0 Then
        mark p
      End If
    End For
    iT := a simple path of marked P_{S_i} nodes
        exists, joining π_M(a) and p_{0_i}
    Return (iT)
End
```

The complexity of this algorithm is O(max{|Ps|+|Ti|, |Pr||Ps|) = O(|Pr||Ti|).

The algorithm to check whether a reachable marking is safe that is, an ordering for the sequential termination of the active processes can be found. -

```
Function isBankerAdmissible
    (In 𝒩: a marked S*PR;
    In M ∈ ℛ𝒮(𝒩, M0)\{M0})
Return (iBA:Boolean)
- -| Pre:
- -| Post: iBA     = Is there a sequential
  termination of the set of
       M-active processes?
Begin
    𝒜 := 𝒜M; M' := M;
    iBA := TRUE; PM' := {p ∈ PS|M(p) > 0}
    While iBA ∧ PM' ≠ ∅
      look for a process a ∈ 𝒜,
      πM'(a) = p ∈ PM',

      s.t. isTerminable     (𝒩, M', a)
     If such a exists Then
      PM' := PM'\{p};
      update M' and 𝒜 assuming all the
        active processes in p teminate
     Else
      iBA := FALSE
     End If
   End While
   Return (iBA)
End
```

The complexity of this algorithm is O((|Ps|^2) |Pr|Ti^max).

CONCLUSION

We've learnt two main concepts from this paper. Firstly, the concept of a class of Petri nets called S*PR. This class is a natural extension of previous classes of nets used in the literature. These nets model RAS with routing flexibility in the processing of parts and multiset of resources at each processing step of a part. The second concept we've learnt about is Banker's-like algorithm for deadlock avoidance in SPR nets and the cost of applying it to decide whether a state is safe or not.