UNIFIED FORMULATION AND COMPARISON OF PULL CONTROL POLICIES USING (MIN,+) ALGEBRA

Jean-Marc BOLLON and Maria DI MASCOLO

Laboratoire d'Automatique de Grenoble (INPG, CNRS, UJF)
LAG-ENSIEG BP.46 38402 Saint-Martin d'Hères Cedex
Jean-Marc.Bollon@inpg.fr, Maria.Di-Mascolo@inpg.fr,
Phone : 33 (0)4 76 82 64 16 Fax : 33 (0)4 76 82 63 88

***Abstract**: In this paper we propose a unified formulation used to express pull control policies. This formulation is given for basestock, kanban, extended kanban and generalized kanban and is then used for the comparison between policies.*

***Keywords**: Pull policies, kanban, basestock, generalized kanban, extended kanban (min,+) algebra, queuing systems.*

1 Introduction

This paper is concerned with make-to-stock pull control policies. In most cases, such systems are decomposed into stages. Each stage i consists of an output buffer, denoted by P_i, which contains the finished parts of the stage, and a manufacturing process, denoted by F_i, which is used to supply the output buffer of the stage. We consider here the case of stages in series (see figure 1) and we assume that all the manufacturing processes consist of a queue with infinite capacity, followed by some stations. In each manufacturing process, we use a push control, but the coordination between the different stages is achieved using a pull mechanism.

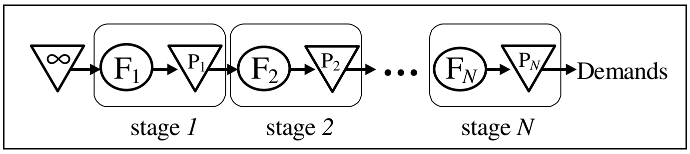

Fig. 1: Production system decomposed into N stages.

A pull mechanism can be implemented in several ways. The best known is the kanban policy (Monden (83)). Another policy, very easy to implement, is the base stock policy (Lee and Zipkin (1992)). Two general control policies, known as the generalized kanban policy (GKS) (Buzacott (1989)), and the extended kanban system (EKS) (Dallery and Liberopoulos (2000)) can also be used to implement the pull mechanism. They are both a combination of kanban and base stock policies.

In this paper, we propose a unified formulation of these four policies. This formulation is obtained using (min,+) algebra, and enables to derive some properties of each policy, and to compare the policies in order to find the set of all the identities of behavior that can exist among these policies.

This paper is organized as follows. In Section 2, we briefly describe the four policies under study. Then, in Section 3, we present our formulation. In Section 4, we illustrate the use of this formulation for the comparison of the policies. Finally, in section 5 we present a generalization of the formulation to other pull control policies.

2 Brief description of the policies

2.1 Base stock system (BSS)

This policy contains one design parameter per stage (and for each type of product): the maximal number of finished parts in a stage i, S_i. The purpose of a BSS is to satisfy demands and to lead each buffer P_i to its maximum level S_i. This policy is very reactive: when an external demand arrives, the information is immediately transferred to all the stages. However, the disadvantage of this method is that the maximum number of work-in-process (WIP) is not limited.

2.2 Kanban system (KS)

This policy has not the same disadvantage as the basestock policy, since a fixed number of cards (or kanbans), K_i, is associated to each stage i of the system, allowing thus to limit the maximum level of WIP and finished parts of the stage. The purpose of a KS is to enable the processing of a new part to begin only when a part is consumed. In a KS, a kanban is attached to each part. When a part at a given stage is consumed by the succeeding stage, the corresponding kanban is detached and can be attached to a new part at the preceding stage, which allows the transfer of information from downstream to upstream of the production system. This policy is thus less reactive than the BSS, since the information is transmitted upstream only when there is a consumption of a part. However, it guarantees a limited number of parts in each stage.

2.3 Generalized kanban system (GKS)

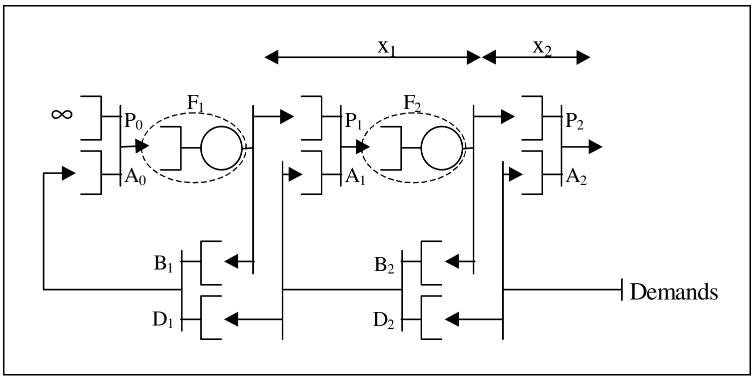

Fig. 2: Queuing network model of a GKS with two stages

A Generalized Kanban control (GKS) is a kind of combination of KS and BSS, which includes these two policies as special cases (the queuing network of figure 2 illustrates the behavior of a GKS composed of two stages). It uses two parameters for each stage i of the production system (and for each type of product). One of the parameters, K_i, limits the maximum level of WIP in the stage, and the other, S_i, determines the maximum number of finished parts in the stage (as for the BSS).

Each arriving demand generates a production order, in order to maintain buffers P_i to their nominal level S_i, as for the BSS. However, these production orders are not transmitted instantaneously upstream (as it was the case for the BSS) since they must be associated to a kanban to go upstream and can thus be stopped because of a lack of kanbans. A part can enter a stage only if there is a kanban available.

The use of two parameters per stage makes the GKS a significant improvement over the KS and the BSS. On the one hand, it loosens the coupling between the transfer of production authorizations and demands, and the release of parts, present in the KS, which enables an efficient transfer of information from the customer to the upstream stages. On the other hand, the control of inventory level is enabled, which was not the case for the BSS.

2.4 Extended Kanban (EKS)

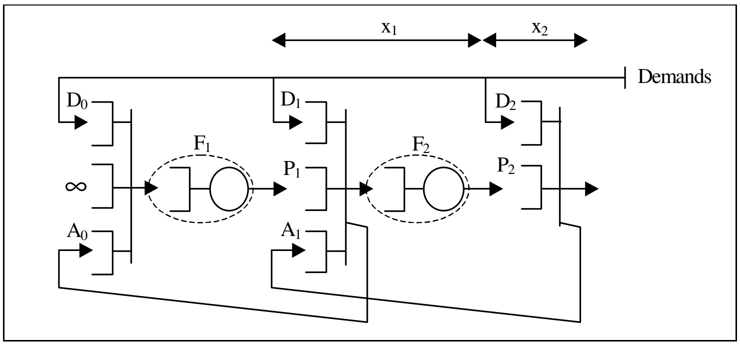

Fig. 3: Queuing network model of an EKS with two stages

This policy is also a combination of BSS and KS and includes these two policies as special cases. Each arriving demand generates a production order for each stage *i* in order to maintain a level S_i for the finished parts of the stage. However, the number of parts in a stage *i* is limited by K_i kanbans. A part can enter a stage only if there are a production order and a kanban available. The queuing network of figure 3 illustrates the behavior of an EKS with two stages.

3 Formulation of Policies

3.1 Some reminding about $(min,+)$ algebra.

We recall here some notions about the (min,+) algebra and its use (see (Bacelli et al. (1992) for more details.

The (min,+) algebra and the (max,+) algebra are often used to model the behavior of timed event graphs. These graphs are composed of places and transitions. The places must have no more than one input and one output. They are also timed with a minimum deterministic delay.

Then two kinds of variables are used to describe the system, both are associated with transitions. The first ones are counters: they count at a given time the number of tokens that have been going through a transition. The other ones are timers which give the moment when a transition is fired for a certain number of times. At the beginning, all the timers and counters have the value zero.

With counters and (min,+) algebra, inequalities can be written to describe the dynamic of the system. Similarly, an equivalent dual system of inequalities can be given with timers and (max,+) algebra. In this paper we only use (min,+) algebra.

The set $\mathbb{R} \cup \{+\infty\}$ associated with the min operator and the usual addition has a dioid algebraic structure named $\mathbb{R}_{\min}$. Then, the operators $\oplus$ and $\otimes$ will stand respectively for the min and the usual addition. The neutral element for $\oplus$ is the infinity and its notation is ε. For $\otimes$ the neutral element is zero and it will be named *e*. Some similar notation can be given with the (max,+) algebra.

For the special case of a timed event graph where every delay on transitions are zero, if transitions are fired at the earliest, then it is possible to express every counter function with input counters and the initial number of token of each places. If input counters are expressed with the components of a vector $u(t) \in \mathbb{R}^m_{\min}$ and other counters by the components of a vector $x(t) \in \mathbb{R}^n_{\min}$ then $x(t)$ is the solution of a following system of inequalities:

$$x(t) \leq a\,x(t) \oplus b\,u(t) \tag{1}$$

a and b are matrixes which belong respectively to $\mathbb{R}^{n\times n}_{\min}$ and $\mathbb{R}^{n\times m}_{\min}$. The smallest solution of the system (1) can be given by:

$$x(t) = a^* b\,u(t) \quad \text{with}^{1} \quad a^* = \bigoplus_{k\in\mathbb{N}} a^k \tag{2}$$

Then the value of $x_i(t)$ can also be similarly given by the minimum value of a sum adding $u_j(t)$ and the shortest oriented path starting from u_j and ending at x_i. This path is searched in a graph where vertices correspond to transitions and arcs correspond to places. Near a place there are two transitions, one before and one after, the vertices related to them are respectively the beginning and the ending of the arc associated with this place. The initial number of tokens present in the place will be the weight of the arc. The control mechanisms in a make-to-stock pull control policies can be expressed by an equivalent event graph, then (min,+) algebra will be useful to describe the policies.

3.2 Elements and notations needed for our formulation of policies

Structure of the system and needed counters. A production line using a pull control policy can be divided into two parts. The first part is composed of stocks P_i and manufacturing processes F_i where the production flow is going down. Each stock is connected to a synchronization station on which acts the second part of the system. This second part is the mechanism of the pull policy. It controls the production requirement flow of information going upstream in the line.

In the production part of the system we use a counter at the end of each manufacturing process and each buffer of parts (see figure 4). The first ones, named u_i, detect the outgoing of products from the manufacturing process and the others, named v_i, count the release of available parts from stocks. If a synchronization station coincides with a counter u_i or v_i then we respectively name it u_i or v_i. In the control part of the system, an input counter d is present to detect the arrival of demands. The pull control policy may request additional synchronization stations w_i that we associate to a counter named w_i.

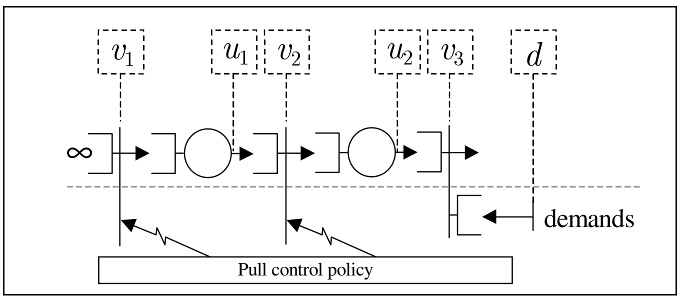

Fig. 4 Main counters of a two stage production line

[1] The star operator is sometimes named “Kleene star”.

For a N stage line, an arc is going from d to a queue D_N at the synchronization station v_N. According to the policy, some other arcs are going from a transition d, $v_{i,}$ u_i or w_i to another one upstream that is v_j or $w_{j.}$

A queuing control state vector X. With the considered policies the state of the system can be expressed with a state vector X used for queuing control problem or a Markov decision process (Veatch and Wein, 1994). This vector X is defined as follows:

$$X_0 = \infty\,, X_N = P_N - D_N \text{ and } X_i = P_i + F_{i+1} \text{ for } i \text{ from 1 to } N\text{-}1$$

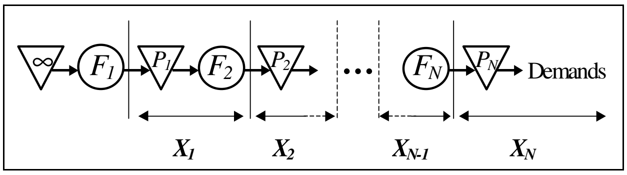

Fig. 5: State variable for an N *stages line*

These components are illustrated by figure 5. A component X_i is equal to the sum of parts present between two consecutive manufacturing outputs. When the vector X_N is positive it represents the number of available finished goods and when it is negative, it is the number of non satisfied demands.

The vector X gives the size of the different inventory positions. These ones are always positives except when they represent the finished goods minus demands.

Every occurrence of an event (an outgoing part from a manufacturing process or a demand arrival) changes the vector X. Conversely, if a vector change is known, then the occurred event is known too. Proofs about the sufficiency of vector X to describe policies under certain conditions are given in (Bollon, 2001).

Knowing that state vector X changes according to the time, we can write the following equalities:

$$X_i(t) - X_i(0) = u_i(t) - u_{i+1}(t) \quad \text{and} \quad X_N(t) - X_N(0) = u_N(t) - d(t) \tag{3}$$

These relations link state vectors to some of the counters.

***The function F_i*(X).** For a majority of cases, the amount of parts present in a manufacturing process given as a function of X is sufficient to describe a pull control policy (Bollon, 2001). Every state vector change allows knowing when an event occurs. When F_i is a function of X it also allows knowing if the policy releases a part in $F_{i.}$ Some parts are released in F_i if and only if $(F_i + X_i)$ or F_i grows.

To calculate the $F_i(X)$ function we use the previous defined counters:

$$F_i(X(t)) = F_i(X(0)) + v_i(t) - u_i(t) \tag{4}$$

Considering that the control part of the system is an event graph and that each $u_i(t)$ and $d(t)$ are inputs of this event graph, then $v_i(t)$ can be substituted by a (min,+) algebra linear expression of $u_i(t)$ and $d(t)$. With equations (3) counters $u_i(t)$ and $d(t)$ can be suppressed and replaced with an expression using $X(t)$. Let us have several examples with the policies basestock, kanban, extended kanban and generalized kanban.

3.3 Calculation of the $F_i(X)$ function for different policies

For these calculations we use the (min,+) algebra notations.

Calculation of $F_i(X)$ for the basestock. In equation (4) we need to evaluate the counter $v_i(t)$. The associated synchronization station v_i is connected to the buffer P_{i-1} with the initial value S_{i-1} , it is also connected to a queue initially empty. An arc is going from d to this queue. Therefore, we have:

$$v_1(t) = d(t) \quad \text{and} \quad v_i(t) = S_{i-1} \otimes u_{i-1}(t) \oplus d(t) \tag{5}$$

Knowing that F_i is initially empty and using (3), (4) and (5) we get:

$$F_1(t) = \bigotimes_{k=1}^{N} S_k / \bigotimes_{k=1}^{N} X_k(t) \quad \text{and} \quad F_i(t) = X_{i-1}(t) \oplus \bigotimes_{k=i}^{N} S_k / \bigotimes_{k=i}^{N} X_k(t) \tag{6}$$

Calculation of $F_i(X)$ for the kanban. The synchronization station v_i is connected to the buffer P_{i-1} with the initial content K_{i-1}, it is also connected to a queue initially empty. An arc is going from v_{i+1} to this queue. After a recursive calculus, we have the following result:

$$v_1(t) = v_2(t) \quad \text{and} \quad v_i(t) = \bigoplus_{k=i}^{N+1} \left(K_{k-1} \otimes u_{k-1}(t) \right) \oplus d(t) \tag{7}$$

The processes F_i are initially empty and $X_i(0) = K_i$. Using (3), (4) and (7) we get:

$$F_1(t) = K_1 \bigoplus_{k=2}^{N} \left(\frac{\bigotimes_{l=1}^{k} K_l}{\bigotimes_{l=1}^{k-1} X_l(t)} \right) \oplus \frac{\bigotimes_{k=1}^{N} K_k}{\bigotimes_{k=1}^{N} X_k(t)} \quad \text{and} \quad F_i(t) = K_i \bigoplus_{k=i+1}^{N} \left(\frac{\bigotimes_{l=i}^{k} K_l}{\bigotimes_{l=i}^{k-1} X_l(t)} \right) \oplus \frac{\bigotimes_{k=i}^{N} K_k}{\bigotimes_{k=i}^{N} X_k(t)} \oplus X_{i-1}(t) \tag{8}$$

Calculation of $F_i(X)$ for the extended kanban. This policy is illustrated in figure 3. On the synchronization station v_i three queues are connected. The first one is the buffer P_{i-1} with the initial content S_{i-1}, the second one is as for the BSS and the last one is as for the KS but it is not empty, it has $K_{i+1} - S_{i+1}$ kanbans. After a recursive calculus, we have the following result:

$$\begin{aligned} v_1(t) &= \bigoplus_{k=1}^{N} \left(\bigotimes_{j=1}^{k} \frac{K_j}{S_j} \otimes \left(S_k \otimes u_k(t) \oplus d(t) \right) \right) \oplus d(t) \\ v_i(t) &= \bigoplus_{k=i-1}^{N} \left(\bigotimes_{j=i}^{k} \frac{K_j}{S_j} \otimes \left(S_k \otimes u_k(t) \oplus d(t) \right) \right) \end{aligned} \tag{9}$$

The processes F_i are initially empty, $X_i(0) = S_i$ and $S_i \le K_i$. Using (3), (4) and (9) we get:

$$\begin{aligned} F_1(t) &= \bigoplus_{k=2}^{N}\left(\bigotimes_{j=1}^{k-1}\left(K_j / X_j(t)\right) \otimes K_k\right) \oplus \bigotimes_{k=1}^{N}\left(S_k / X_k(t)\right) \oplus K_1 \\ F_i(t) &= \bigoplus_{k=i+1}^{N}\left(\bigotimes_{j=i}^{k-1}\left(K_j / X_j(t)\right) \otimes K_k\right) \oplus \bigotimes_{k=i}^{N}\left(S_k / X_k(t)\right) \oplus K_i \oplus X_{i-1} \end{aligned} \tag{10}$$

Calculation of $F_i(X)$ for the generalized kanban. For this policy, we use the notations illustrated on the queuing network of figure 2. The synchronization station v_i is connected to the buffer P_{i-1} with the initial content S_{i-1}, it is also connected to a queue A_{i-1} initially empty. In the control part of the production system an arc is going from a synchronization station w_i to A_{i-1}. On w_i the queues B_i and D_i are respectively fed with an arc coming from u_i and w_{i+1}. After a recursive calculus, we have the following result:

$$v_1(t) = d(t) \bigoplus_{j=1}^{N} K_j \cdot u_j(t) \quad \text{and} \quad v_i(t) = d(t) \bigoplus_{j=i}^{N} K_j \cdot u_j(t) \oplus S_{i-1} \cdot u_{i-1}(t) \tag{11}$$

The processes F_i are initially empty and $X_i(0) = S_i$. Using (3), (4) and (11) we get:

$$\begin{aligned} F_1(t) &= \bigoplus_{j=2}^{N}\left(K_j \otimes \bigotimes_{k=1}^{j-1}\left(S_k / X_k(t)\right)\right) \oplus \bigotimes_{k=1}^{N}\left(S_k / X_k(t)\right) \oplus K_1 \\ F_i(t) &= \bigoplus_{j=i+1}^{N}\left(K_j \otimes \bigotimes_{k=i}^{j-1}\left(S_k / X_k(t)\right)\right) \oplus \bigotimes_{k=i}^{N}\left(S_k / X_k(t)\right) \oplus K_i \oplus X_{i-1} \end{aligned} \tag{12}$$

Summary results leading to a canonical formulation. For the BSS, KS, EKS and the GKS the function $F_i(X)$ have always the following structure:

$$F_1(t) = \bigoplus_{j=0}^{N}\left(C_{(1,j)} / \bigotimes_{k=1}^{j} X_k(t)\right) \quad \text{and} \quad F_i(t) = \bigoplus_{j=i-1}^{N}\left(C_{(1,j)} / \bigotimes_{k=i}^{j} X_k(t)\right) \oplus X_{i-1} \tag{13}$$

This formulation will be called the canonical formulation. The parameters $C_{(i,j)}$ of this formulation define the policy.
For a BSS we have:

$$C_{(i,N)} = \bigotimes_{k=i}^{N} S_k \text{ and } C_{(i,j)} = \varepsilon$$

for the KS:

$$C_{(i,N)} = \bigotimes_{k=i}^{N} K_k \text{ and } C_{(i,j)} = \bigotimes_{k=i}^{j+1} K_k$$

for the EKS:

$$C_{(i,N)} = \bigotimes_{k=i}^{N} S_k \text{ and } C_{(i,j)} = \bigotimes_{k=i}^{j+1} K_k$$

and for the GKS:

$$C_{(i,N)} = \bigotimes_{k=i}^{N} S_k \text{ and } C_{(i,j)} = K_{j+1} \otimes \bigotimes_{k=i}^{j} S_k$$

with for each parameter $i-1 \le j < N$.

The state space. A state X is admissible if and only if every $F_i(X)$ is non negative. Then the state space for a canonical formulation is given by the following polyhedron

$$\sum_{k=i}^{j} X_k \le C_{(i,k)} \tag{14}$$

with $1 \le i \le j \le N$ and $X \in \mathbb{N}^{N-1} \times \mathbb{Z}$.

4 Comparison between policies

This formulation can be used to discover functioning identities between two policies. Especially, we found every possible parameter values for which EKS is similar to GKS (Bollon 2001). These results are summarized below, in sections 4.1 and 4.2.

When manufacturing processes contain only one server, it suffices to look for the identity of equations (14) to compare two policies. For more general manufacturing processes two policies have an identical behavior only if all the $F_i(X)$ functions are equal.

To compare functions $F_i(X)$ we must look at the arguments of the min function. However, some arguments do not have to be examined if they are too great.

It is also possible to compare a policy to itself using different parameters. This way we found some results previously discovered for the GKS.

If each function $F_i(X)$ with one given policy is lower than the one obtained with another policy, then the production capacity is lower for the first one. For example with similar parameter EKS production capacity is higher than GKS production capacity (Dallery and Liberopoulos, 2000).

4.1 Comparison between two-stages EKS and GKS.

Single server case. Let us consider a two stages system whose manufacturing processes are composed of a queue followed by a single server. We proved (Bollon 2001) that both policies EKS and GKS are identical for this system if and only if (exponents e and g are used to make the difference between the EKS and the GKS parameters):

$$S_2^g = S_2^e,\ S_1^g = S_1^e \text{ and } k_2^g + S_1^g = k_2^e + k_1^e$$

Moreover, for any EKS it is possible to find an equivalent GKS whose parameters are functions of the EKS parameters:

$$S_2^g = S_2^e,\ S_1^g = S_1^e, k_2^g = k_2^e + k_1^e - S_1^e \text{ and } k_1^g \text{ can take any value.}$$

Finally, it is possible to define an EKS equivalent to a GKS if and only if $S_2^g \le k_2^g$. Then, we can determine the parameters of the EKS from the parameters of the GKS:

$$S_2^e = S_2^g,\ S_1^e = S_1^g,\ k_1^e = k_2^g + S_1^e - k_2^e \text{ and } S_2^g \le k_2^e \le k_2^g$$

Some similar results are observed in (Karaesmen and Dallery 2000).

General manufacturing processes. Now, the conditions for the equivalence between EKS and GKS are more restrictive. These two systems are equivalent if and only if:

$$k_1^g = S_1^g = k_1^e = S_1^e \text{ , } S_2^g = S_2^e \text{ and } k_2^g = k_2^e .$$

That means that a kanban policy is used for the first stage.

4.2 Comparison between multiple-stages EKS and GKS.

Single server case. When we consider a production system composed of N stages, each stage consisting in a single server, it can be proved (Bollon 2001) that, for a given EKS, we can find an equivalent GKS only if the stages that are upstream stages N and N-1 are controlled by a kanban policy (i.e. $S_j^g = k_j^g$ for $j = 1, \ldots, N\text{-}2$)

General manufacturing processes. In this case, we will have an equivalence between an EKS and a GKS if and only if a kanban policy is used for the N-1 upstream stages ($S_j^g = k_j^g = k_j^e = S_j^e$ for $j = 1,\ldots,N\text{-}1$) and the two policies have the same parameters ($k_N^g = k_N^e$), which implies $S_N^g \le k_N^g$.

5 A generalized use of the canonical formulation

In this section, we show that it is possible to express a category of pull policy with our canonical formulation. This category is defined in the proposition 1 and illustrated in the figure 6.

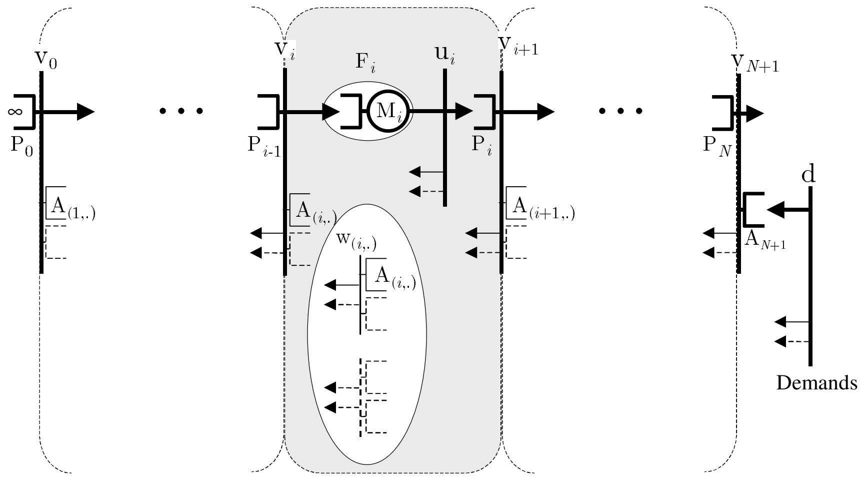

Fig. 6 Description of the policy category having a canonical formulation.

Proposition 1 :

A queuing production system has a canonical formulation $F_i(X)$ if it is defined as following.

♦ The queuing system is composed in its production part of :

- Manufacturing processes F_i including a queue followed by a station M_i .
- Synchronization stations u_i (associated with the counter $u_i(t)$) having a unique input coming from M_i and an output going to the queue P_i .

- Stocks with queues P_i. The raw parts queue P_0 is never empty.
- Synchronization stations v_{i+1} (associated with the counter $v_{i+1}(t)$) having an input coming from P_i and an output going to F_{i+1} or out for $i = N$.

♦ In the control part, the system is composed of :

- Synchronization stations $\mathrm{w}_{(i,j)}$ (associated with the counter $w_{(i,j)}(t)$) with outputs going to some queues $\mathrm{A}_{(l,k)}$, where $1 \le l \le i \le N$.
- Queues $\mathrm{A}_{(i,j)}$ linked to v_i or $\mathrm{w}_{(i,k)}$ and fed with u_k, v_{k+1}, $\mathrm{w}_{(k,l)}$ or d where $1 \le i \le k \le N$.
- A queue A_{N+1} linked to v_{N+1} and fed with d. ❑

This proposition can be proved recursively from downstream stages to upstream stages (Bollon 2001). The queuing network in the control part is evaluated by a solution given with (2). We also derived an algorithm able to calculate the canonical formulation for any policy consistent with the proposition 1. The core of this algorithm is a shortest path search.

6 Conclusion

Pull control policies (most of them) are described with a single simple formulation. A computing algorithm for this formulation can be obtained when manufacturing lines are expressed with queuing networks.
This formulation allows to find any similar working between two policies and may also enable to find some special properties.
We expect, with this formulation, to find some more general results.

References

F. Baccelli, G. Cohen, G.J. Olsder and J.P. Quadrat (1992) *Synchronization and Linearity: An Algebra for Discrete Event Systems*. John Wiley and Sons, NewYork.

Bollon J-M. (2001). Etude de différentes politiques de pilotage de systèmes de production. *PhD Thesis,* LAG - INPG.

Buzacott J. A. (1989). Queueing models of Kanban and MRP controlled production systems. *Engineering Cost and Production Economics*, 17.

Dallery Y. and Liberopoulos G. (2000). Extended kanban control system: combining kanban and base stock. *IIE Transactions,* **32***,* 369-386

Karaesmen F. and Dallery Y. (2000) A performance comparison of pull type control mechanisms for multi-stage manufacturing. *International Journal of Production Economics*, **68**, N°1.

Lee Y.J. and Zipkin P., (1992) "Tandem queues with Planned Inventories", *Operations Research*, **40**, N°5.

Monden Y. (1983). *Toyota Production System*. Industrial Engineering and Management Press.

Veatch M. H. and Wein L. M. (1994). Optimal control of a two-station tandem production/inventory system - *Oper. Res.* **42**, No.2, 337-350.