IEEE Int. Conf. on Systems, Man and Cybernetics (SMC'06), Beijing, China, 14-17 October 1996, pp.2990-2995.
 (DOI 10.1109/ICSMC.1996.561440).

Composite Schedules of Manufacturing Cells and Their Timed Petri Net Models

W.M. Zuberek

Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada A1C–5S7

A b s t r a c t

In composite schedules, several (identical or different) parts enter and leave a manufacturing cell in each cycle. A systematic method of generating all composite schedules is proposed, and it is shown that the generated schedules can easily be transformed into timed Petri net models. Invariant analysis of these timed net models provides performance characteristics of the cell. The characteristics are obtained in analytical (or symbolic) form, so they are applicable to a wide spectrum of specific cases. Simple examples illustrate an application of the proposed approach to a robotic cell with three machines.

1. INTRODUCTION

In flexible manufacturing systems, machines are often grouped into manufacturing cells (or robotic cells), in which a robot performs cyclic sequences of pickup, move, load, unload and drop operations, transporting the manufactured parts from one machine of the cell to another [4, 5, 2]. This cyclic sequence of robot activities is called a schedule. The schedule determines the throughput of the cell, because it determines the sequence of robot's actions as well as the sequence in which the parts enter the cell [1]. Any approach to maximizing the throughput of a robotic cell must thus deal efficiently with two issues: how to generate alternative schedules for a given cell, and how to evaluate these schedules.

Two types of schedules can be identified for manufacturing cells, the so called simple schedules, in which exactly one part enters the cell and one leaves the cell in each cycle (although the leaving part may not be the same as the entering one), and composite schedules, which deal with several (new) parts in each cycle. The case of simple schedules was previously discussed in detail [7]. Models of composite schedules can be regarded as a composition of simple schedules, and each such simple schedule corresponds to one part entering the cell within each cycle. A systematic generation of composite schedules is proposed in this paper; the schedules are generated by systematically tracing all possible moves of parts through the cell. Moreover, it is shown how each composite schedule can be converter into a timed Petri net model and then evaluated.

In timed Petri net models, the duration of modeled activities is also taken into account and included into model specifications. Several types of Petri nets 'with time' have been proposed by assigning 'firing times' to the transitions or places of a net. In timed nets, transition firings are 'real–time' events, i.e., tokens are removed from input places at the beginning of the firing period, and they are deposited to the output places at the end of this period (sometimes this is also called a "three–phase" firing mechanism). The firing times may be either deterministic or stochastic, i.e., described by some probability distribution function. In both cases the concepts of state and state transitions have been formally defined and used in derivation of different performance characteristics of the model [6].

In modeling manufacturing cells, the structure of the net models represents the "flow" of parts through the cell and the sequence of robot's operations, while the times associated with transitions describe the (average) operation times of machines of the cell as well as the robot. Consequently, there is a close correspondence between the elements of the model and the modeled system.

Analysis of net models can be based on their behavior (i.e., the space of reachable states) or on the structure of the net; the former is called reachability analysis while the latter structural analysis. Invariant analysis seems to be the most popular example of the structural approach. Structural methods eliminate the derivation of the state space, so they avoid the 'state explosion' problem of reachability analysis, but they cannot provide as much information as the reachability approach does. Quite often, however, all the detailed results of reachability analysis are not really needed, and more synthetic performance measures, that can be provided by structural methods,

are quite satisfactory [2]. In particular, the throughput (or the cycle time) of a timed net model can easily be determined from the structure of a net [7].

This paper presents a systematic method of generating all composite schedules for a given cell. Each of such composite schedules is modeled by a timed Petri net, and its performance is determined by invariant analysis of the derived net model. Simple examples of manufacturing cells are used as an illustration of the proposed approach.

2. SIMPLE SCHEDULES

Simple schedules of robotic cells are schedules in which exactly one part enters and one leaves the cell in each cycle (although the part which leaves the cell may not be the same as the one which enters the cell). It is known [4] that for a cell with m machines, there are $m!$ different simple schedules. For $m = 3$ (a sketch of a 3–machine cell is shown in Fig.1) there are six simple schedules, which are denoted here as A, B, C, D, E and F.

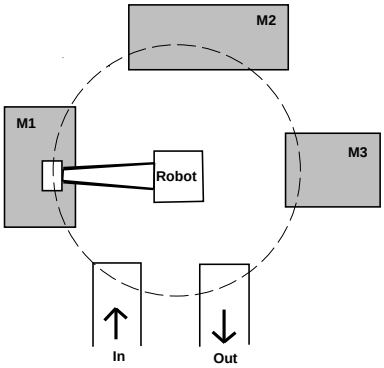


Fig.1. Layout of a three–machine cell.

Assuming, for simplicity, that each part follows the same path from the input (In) to machine–1 (M_1), to machine–2 (M_2), to machine–3 (M_3), and finally to the output of the cell (Out), the simple schedules can be described by the following sequences of cell configurations, where each configuration corresponds to a distribution of parts among the machines of the cell (when the robot does not carry a part); more specifically, for an m–machine cell, each configuration is described by an m–tuple of machine descriptions:

$$(k_1, k_2, ..., k_m)$$

where each machine description k_i is "1" if the machine M_i is loaded with a part in this configuration, and otherwise is "0" (in the case of multiple machines performing exactly the same operations, the values describing each multi–machine station would assume the values from "0" to "n" where n is the number of identical machines). The six simple schedules for a 3–machine cell are:

A: $(0,0,0) \to (1,0,0) \to (0,1,0) \to (0,0,1) \to (0,0,0)$
B: $(0,0,1) \to (1,0,1) \to (0,1,1) \to (0,1,0) \to (0,0,1)$
C: $(0,0,1) \to (1,0,1) \to (1,0,0) \to (0,1,0) \to (0,0,1)$
D: $(0,1,0) \to (1,1,0) \to (1,0,1) \to (1,0,0) \to (0,1,0)$
E: $(0,1,0) \to (1,1,0) \to (1,0,1) \to (1,0,0) \to (0,1,0)$
F: $(0,1,1) \to (1,1,1) \to (1,1,0) \to (1,0,1) \to (0,1,1)$

Each change of configurations corresponds to a part moving from one machine to another, from the input to the first machine, or from the last machine to the output; all schedules uniformly begin by moving a (new) part from the input to the first machine.

The simple schedules can be generated systematically by tracing all possible "passages" of parts through the cell [8]. These passages are correspond to the following changes of cell configurations:

- a configuration $(k_1, ..., k_i, k_{i+1}, ..., k_m)$ derives a configuration $(k_1, ..., k_i - 1, k_{i+1} + 1, ...k_m)$ if and only if the value of k_i is "1" and the value of k_{i+1} is "0", $i = 1, ..., m-1$;
- a configuration $(k_1, k_2, ..., 1)$ always derives a configuration $(k_1, k_2, ..., 0)$ (this derivation corresponds to moving a part from the last machine M_m to the output of the cell),
- it is assumed that each schedule begins by moving a (new) part from the input to the machine M_1, so the first derivation is always from $(0, k_2, ...k_m)$ to $(1, k_2, ..., k_m)$,
- for a cell with m machines, the length of all simple schedules is equal to $m + 1$ (it corresponds to a passage of a part, although not necessarily the same, from the input, through all machines of the cell, to the output).

Since parts are transported from one machine (or input) to another (or output) by the robot, the sequences of robot's actions can easily be derived from the sequences of configurations by "implementing" the moves of parts corresponding to changes of consecutive configurations. For example, schedule A begins be transporting a part from the input to M_1 and loading it; when the first operation is finished, the robot unloads M_1, moves the part to M_2 and loads it there, and so on. The sequences of robot actions are as follows (the robot moves from X to Y are denoted by $X \Rightarrow Y$ if the robot carries a part and by $X \to Y$ otherwise):

A: $In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \to In$
B: $In \Rightarrow M_1 \Rightarrow M_2 \to M_3 \Rightarrow Out \to M_2 \Rightarrow M_3 \to In$
C: $In \Rightarrow M_1 \to M_3 \Rightarrow Out \to M_1 \Rightarrow M_2 \Rightarrow M_3 \to In$
D: $In \Rightarrow M_1 \to M_2 \Rightarrow M_3 \to M_1 \Rightarrow M_2 \to M_3 \Rightarrow Out \to In$
E: $In \Rightarrow M_1 \to M_2 \Rightarrow M_3 \Rightarrow Out \to M_1 \Rightarrow M_2 \to In$
F: $In \Rightarrow M_1 \to M_3 \Rightarrow Out \to M_2 \Rightarrow M_3 \to M_1 \Rightarrow M_2 \to In$

3. COMPOSITE SCHEDULES

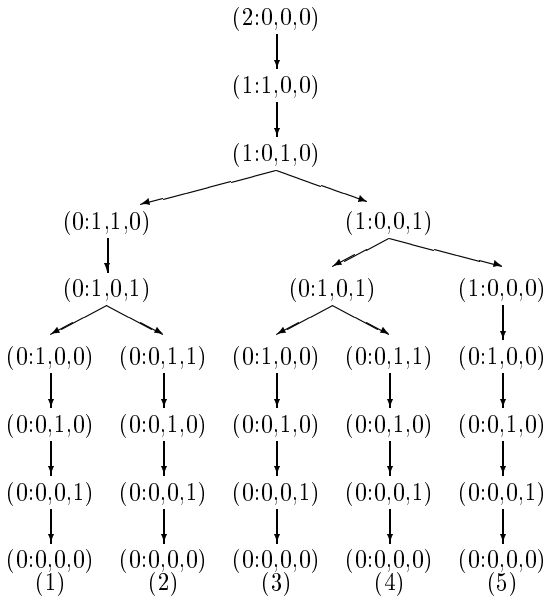
For composite schedules, several parts enter and leave the cell in each cycle. Models of composite schedules can be regarded as interleaved compositions of simple schedules, corresponding to processing of consecutive parts which enter the cell within one cycle, and which move from one machine to another in consecutive steps of the schedule. A systematic generation of composite schedules can be obtained by a simple extension of the procedure used for generation of simple schedules. Composite schedules which process n parts withing a single cycle are called simply n–schedules. For composite schedules, the configurations are extended by a hypothetical "input container", which initially contains exactly n parts for each n–schedule, and which must be emptied during the schedule. This container is represented by an additional element of configurations, for example, the initial element, separated from the remaining machine descriptions by a colon (rather than of a comma). A typical configuration for description of composite schedules of an m–machine cell is thus:

$$(k_0 : k_1, k_2, ..., k_m)$$

The (revised) rules describing changes of configurations are:

- a configuration $(k_0 : k_1, ..., k_i, k_{i+1}, ..., k_m)$ derives a configuration $(k_0 : k_1, ..., k_i - 1, k_{i+1} + 1, ...k_m)$ if and only if the value of k_i is "1" and the value of k_{i+1} is "0", $k = 1, ..., m-1$;
- a configuration $(k_0 : k_1, k_2, ..., 1)$ always derives a configuration $(k_0 : k_1, k_2, ..., 0)$ (this derivation corresponds to moving a part from the last machine M_m to the output of a cell),
- a configuration $(k_0 : 0, k_2, ..., k_m)$ derives a configuration $(k_0 - 1 : 1, k_2, ..., k_m)$ if and only if the value of k_0 is greater than 0 (this derivation corresponds to moving a part from the input to the first machine M_1),
- it is assumed that each schedule begins by moving a (new) part from the input to the machine M_1, so the first derivation is always $(k_0 : 0, k_2, ...k_m)$ to $(k_0 - 1 : 1, k_2, ..., k_m)$,
- for a cell with m machines, the length of all n–schedules is equal to $n * (m + 1)$.

For a 3–machine cell, there are 34 different 2–schedules, including 6 schedules which are just simple schedules repeated twice. All 34 2–schedules can be systematically derived by repeatedly applying the rules to the four initial configurations of the cell. For the initial configuration (0,0,0), there are five 2–schedules:



Each of these schedules can be decomposed into a pair of interleaved simple schedules, for example, schedule (1) is composed of simple schedules A and E:

schedule A		*schedule E*
(0,0,0)		
(1,0,0)		
(0,1,0)	→	(0,1,0)
		(1,1,0)
		(1,0,1)
		(1,0,0)
(0,1,0)	←	(0,1,0)
(0,0,1)		
(0,0,0)		

schedule (4) is a composition of A and B:

schedule A		*schedule B*
(0,0,0)		
(1,0,0)		
(0,1,0)		
(0,0,1)	→	(0,0,1)
		(1,0,1)
		(0,1,1)
		(0,1,0)
(0,0,1)	←	(0,0,1)
(0,0,0)		

and schedule (5) is simply a composition of A with itself.

All these schedules can easily be translated into sequences of robot actions (the robot moves from X to Y are denoted by $X \Rightarrow Y$ if the robot carries a part and by $X \rightarrow Y$ otherwise):

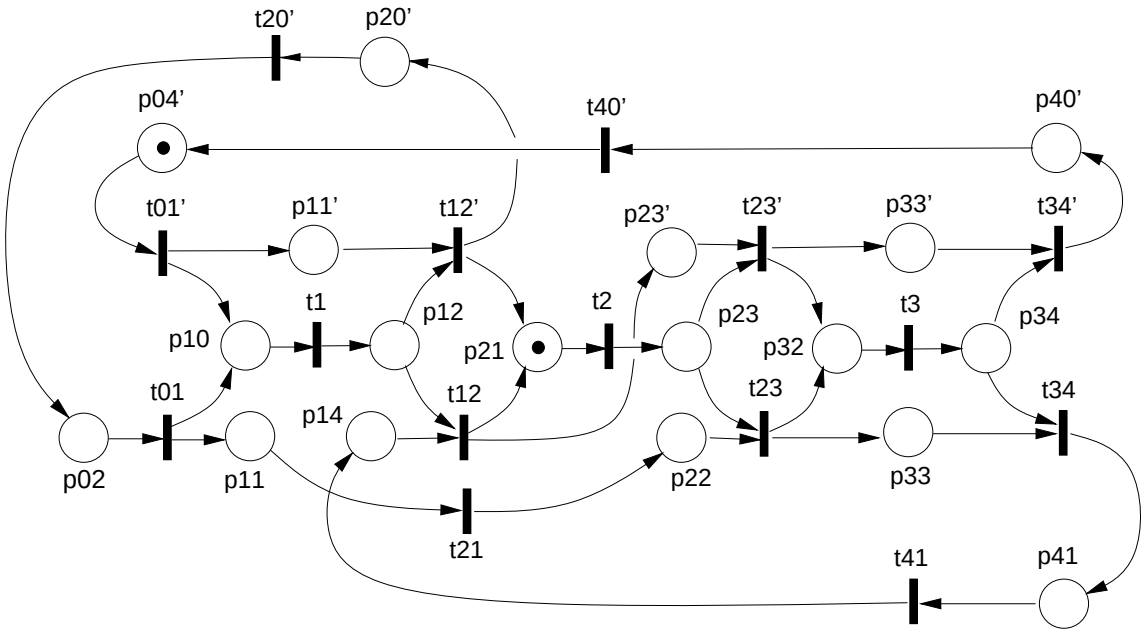


Fig.2. Petri net model of schedule (1)=(A+E).

(1): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

(2): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

(3): $In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

(4): $In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

(5): $In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In$

Timed Petri net models can easily be derived from the sequences of robot's action. In timed models, net transitions represent (machine and robot) operations while net places represent 'conditions' (in the most general sense). For example, the net model for schedule (1), i.e., A+E, is shown in Fig.2. The three machines of Fig.1 (or rather machine operations) are represented by t_1, t_2 and t_3, and each of these transitions has an input and output place (for 'part loaded' and 'machine operation finished' conditions). The 'firing times' associated with these transitions, $f(t_1) = o_1$, $f(t_2) = o_2$ and $f(t_3) = o_3$, represent the (average) times of performing the operations on machines M_1, M_2 and M_3, respectively. The remaining part of the net represents the sequence of robot operations, which follows the sequence of actions indicated above. It is assumed that there is always an available part in In and that Out removes manufactured parts sufficiently quickly, so In and Out are not actually shown (although they can easily be added to the model).

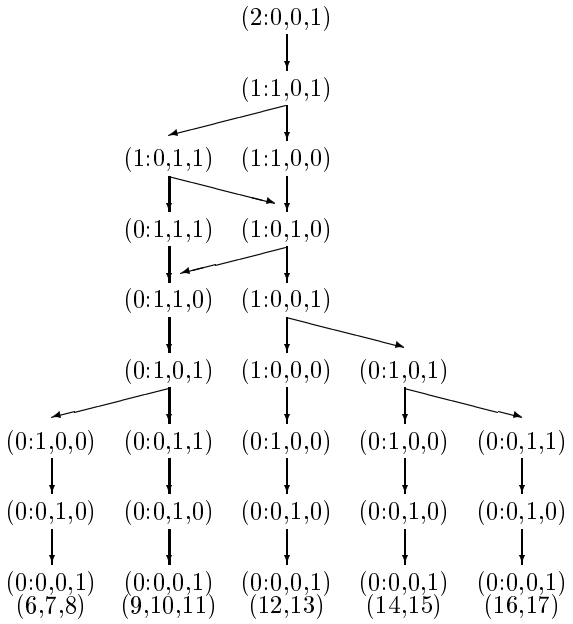
The transitions correspond to the following actions for the A and E parts of the schedule:

	robot operations	*exec time*
t'_{01}	pick from In, move to M_1, load	$u+w+y$
t'_{12}	unload M_1, move to M_2, load	$v+w+y$
t'_{20}	move from M_2 to In	$2y$
t'_{23}	unload M_2, move to M_3, load	$v+w+y$
t'_{34}	unload M_3, move to Out, drop	$v+x+y$
t'_{40}	move from Out to In	y
t_{01}	pick from In, move to M_1, load	$u+w+y$
t_{12}	unload M_1, move to M_2, load	$v+w+y$
t_{21}	move from M_1 to M_2	y
t_{23}	unload M_2, move to M_3, load	$v+w+y$
t_{34}	unload M_3, move to Out, drop	$v+x+y$
t_{41}	move from Out to M_1	$2y$

where the *execution times* (or firing times of transitions) are given assuming that:

u denotes the (average) pickup time,
v – the (average) unload time,
w – the (average) load time,
x – the (average) drop time,
y – the average 'travel' time between two adjacent machines (assuming, for simplicity, that this time is the same for all adjacent machines, and also the same for M_3 to Out, Out to In and In to M_1 moves).

For the initial cell configuration (0,0,1), there are 12 different 2–schedules:



and their corresponding robot's sequences of actions are:

(6): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(7): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(8): $In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(9): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(10): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(11): $In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \rightarrow In \Rightarrow M_1 \rightarrow M_2 \Rightarrow M_3 \rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(12): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(13): $In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \Rightarrow Out \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

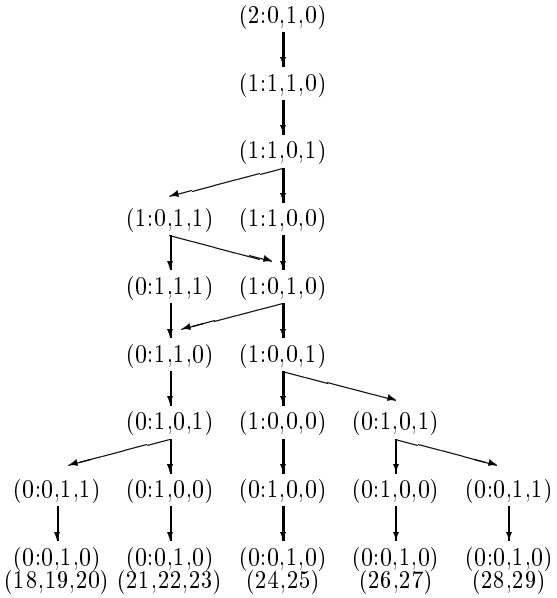
(14): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(15): $In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In$

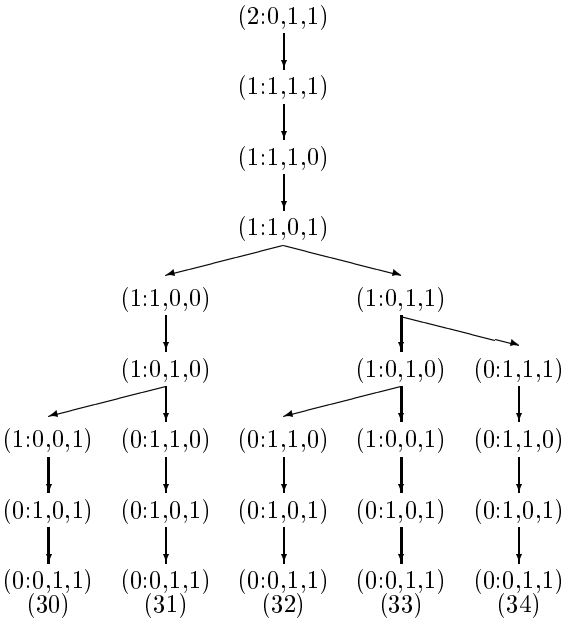
(16): $In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In$

(17): $In \Rightarrow M_1 \rightarrow M_3 \Rightarrow Out \rightarrow M_1 \Rightarrow M_2 \Rightarrow M_3 \rightarrow In \Rightarrow M_1 \Rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \rightarrow M_3 \Rightarrow Out \rightarrow M_2 \Rightarrow M_3 \rightarrow In$

There are 12 schedules for the initial configuration (0,1,0):



and another 5 schedules for the initial configuration (0,1,1):



4. EVALUATION OF SCHEDULES

The cycle times of schedules can be determined using net invariants. For a Petri net $\mathcal{N} = (P, T, A)$ with a connectivity matrix $\mathcal{C}_{k\ell}$ where k is the number of places and ℓ is the number of transitions, a place–invariant (or simply P–invariant) $\mathcal{I}$ is any non–negative, k–element vector which satisfies the matrix equation [3]:

$$\mathcal{C}^t \times \mathcal{I} = 0$$

where $\mathcal{C}^t$ denotes the transpose of matrix $\mathcal{C}$.

It can be observed that any linear combination of P–invariants is also a P–invariant. A basic invariant is any invariant which does not contain any other invariants. The cycle times of basic invariants determine the cycle time of the model.

The net shown in Fig.2 has five basic P–invariants which imply the following subsets of transitions [8] (all entries equal to 2 correspond to P_i–implied subnets in which the corresponding transitions are implied twice):

P–invariant:	1	2	3	4	5
t_1	2	0	0	0	0
t_2	0	2	2	0	0
t_3	0	2	0	2	0
t'_{01}	1	1	1	1	1
t'_{12}	1	1	1	1	1
t'_{20}	1	0	0	1	1
t'_{23}	1	1	1	1	1
t'_{34}	1	1	1	1	1
t'_{40}	1	1	1	1	1
t_{01}	1	0	0	1	1
t_{12}	1	1	1	1	1
t_{21}	0	0	0	1	1
t_{23}	0	1	1	1	1
t_{34}	0	1	1	1	1
t_{40}	0	1	1	1	1

The cycle time is determined by the basic invariant with the maximum cycle time. Since the set of transitions of the P–invariant (3) is a subset of that of (2), and the set of transitions of (5) is a subset of that of (4), the cycle time for this schedule is:

$$\tau_0 \;=\; \max(\tau_1, \tau_2, \tau_4)$$

where:

$$\begin{aligned} \tau_1 &= 2o_1 + 2u + 4v + 5w + 9y, \\ \tau_2 &= 2o_2 + 2o_3 + u + 6v + 5w + 2x + 10y, \\ \tau_4 &= 2(o_3 + u + 3v + 3w + x + 7y) \end{aligned}$$

All other schedules can be evaluated similarly using net invariants. The schedule with the minimal cycle time is the optimal 2–schedule.

5. CONCLUDING REMARKS

A systematic approach to modeling and analysis of composite schedules for a large class of manufacturing cells is proposed. The derived net models are composed of a relatively small number of subnets which can easily be determined by invariant analysis. Performance characteristics (the throughput or the average cycle time) in symbolic form which means that specific values of performances can easily be obtained by evaluating the symbolic results for specific values of parameters (i.e., symbols), and then the best schedule (i.e., the one with the smallest cycle time) can be selected to maximize the cell's performance.

Several simplifying assumptions were made during the derivation of Petri net models, e.g., the all parts are identical, that the robot travel times between adjacent machines are the same, etc. It should be noted, however, that all these assumptions can easily be removed by simple modifications of the presented approach. For example, composite schedules can be used to describe scheduling problems when parts of different types enter and leave the cell in one cycle. Parameters associated with such parts can easily be introduced in the model because the corresponding operations are represented by different transitions.

Acknowledgement

The Natural Sciences and Engineering Research Council of Canada partially supported this research through Research Grant A8222.

R e f e r e n c e s

[1] Dixon, C., Hill, S.D.: "Work–cell cycle–time analysis in a flexible manufacturing system"; Proc. Pacific Conf. on Manufacturing, Sydney–Melbourne, Australia, vol.1, pp.182–189, 1990.

[2] Hillion, H.P.: "Timed Petri nets and application to multi-stage production system"; in: Advances in Petri Nets 1989 (Lecture Notes in Computer Science 424); pp. 281–305, Springer Verlag 1989.

[3] Reisig, W.: "Petri nets - an introduction" (EATCS Monographs on Theoretical Computer Science 4); Springer Verlag 1985.

[4] Sethi, S.P., Sriskandarajah, C., Sorger, G., Blazewicz, J., Kubiak, W.: "Sequencing of parts and robot moves in a robotic cell"; Int. Journal of Flexible Manufacturing Systems, vol.4, pp.331–358, 1992.

[5] Silva, M., Valette, R.: "Petri nets and flexible manufacturing"; in: "Advances in Petri nets 1989" (Lecture Notes in Computer Science 424), pp. 374–417, Springer Verlag 1989.

[6] Zuberek, W.M.: "Timed Petri nets – definitions, properties and applications"; Microelectronics and Reliability (Special Issue on Petri Nets and Related Graph Models), vol.31, no.4, pp.627–644, 1991.

[7] Zuberek, W.M., Kubiak, W., "Timed Petri net models of flexible manufacturing cells"; Proc. 36-th Midwest Symp. on Circuits and Systems, Detroit, MI, pp.922–925, 1993.

[8] Zuberek, W.M.: "Application of timed Petri nets to modeling and analysis of flexible manufacturing cells"; Technical Report #9503, Department of Computer Science, Memorial University of Newfoundland, St.John's, Canada A1C 3X5 (available through anonymous ftp at `ftp.cs.mun.ca` in `pub/techreports` as `tr_9503.ps.Z`).