8-th Int. Workshop on Petri Nets and Performance Models, Zaragoza, Spain, September 8-10, 1999, pp.92-101.
Copyright ©1999 IEEE (DOI 10.1109/PNPM.1999.796556).

Stepwise Refinements of Net Models and Their Place Invariants

W.M. Zuberek
Department of Computer Science
Memorial University of Newfoundland
St.John's, Canada A1B 3X5

Abstract

Schedules for manufacturing cells can be systematically derived by simple stepwise refinements which, in consecutive steps, increase the complexity of the cell by introducing its components one after another. Timed Petri net models of schedules derived in this way have some convenient structural properties – net models are covered by conflict–free subnets, determined by place invariants of the model. These place invariant implied subnets can be used for evaluation of the basic performance characteristics of the model. The paper shows that place invariants of net models of schedules can be obtained by the same stepwise refinements that are used for model derivation. Simple examples of performance evaluation are included as an illustration of the use of place invariants in the analysis of schedules.

1. Introduction

Petri nets [13, 11] have been proposed as a formalism for modeling and analysis of discrete–event systems with asynchronous, interacting components. This includes modeling, control and analysis of automated manufacturing systems [1, 4, 5, 6, 15, 16]. Popularity of net models is due to a simple and 'natural' representation of those aspects of systems that cannot easily be modeled using queueing theory or other traditional modeling and evaluation techniques; concurrent and asynchronous events, typical for many discrete–event dynamical systems, are easily captured by Petri nets. In addition, a well–developed mathematical foundation exists for describing and analyzing net models.

In order to study the performance aspects of Petri net models, the duration of activities must also be taken into account. Several types of Petri nets 'with time' have been proposed by assigning 'firing times' to transitions or 'enabling times' to places. In timed Petri nets [12, 2, 8, 17], the events occur in 'real time', i.e., there is a (deterministic or stochastic) duration associated with each transition's firing, and different (concurrent) firings of transitions correspond to (concurrent) activities in the modeled systems. For timed Petri nets, the concept of 'state' and state transitions can be formally defined, and used to derive different performance characteristics of the model [17].

Petri net models have been used for modeling manufacturing systems and studying their scheduling problems [3, 7]. Scheduling can be dynamic, in which case the decisions about the ordering of operations (which may be in conflict, for example, because of shared resources) are performed 'on line', just before the execution of the operations, or static, in which case the decisions are made during a pre–execution stage, and are represented by a (usually cyclic) fixed sequence of operations that attempts to optimize the performance of the system. Only static scheduling is considered in this paper.

Manufacturing systems are often composed of clusters of machines connected by a transportation system that moves the parts between the clusters of machines and between the machines and storage facilities. Each cluster of machines can be organized in a manufacturing cell (or a robotic cell) [14] in which a robot transports the parts from one machine to another and also from the input conveyor to the first machine, and from the last machine to the output conveyor. The sequence of operations performed (cyclically) by the robot is called the schedule for a cell. The performance of the cell is determined by the ordering of robot's operations, i.e., by the schedule.

It can be shown [19] that schedules for manufacturing cells can be systematically derived by stepwise refinements applied to single elements of net models (transition refinements). It is also known that, for some classes of nets, basic performance characteristics (such as throughput or cycle time) can be determined from structural properties of nets. More specifically, if a net is covered by a family of conflict–free cyclic subnets, determined by basic place invariants of the net, the cycle time of the net is equal to the maximum cycle time of the invariant–implied subnet. The evaluation of performance can thus be based on the set of basic place invariants of the net model.

This paper describes an approach to deriving place invariants of net models by stepwise refinements. The approach is used for performance analysis of schedules for manufacturing cells. Since the analyzed schedules are also derived by stepwise refinements, a combined approach is proposed in which both the model and its performance are derived by a sequence of transition refinements, starting from a trivial initial model. The refinements preserve the boundedness, liveness, absence of deadlocks, and many other structural properties of the model.

Section 2 recalls some basic concepts related to manufacturing cells and their schedules. Fundamental elements of Petri nets and timed nets, their place invariants, transition refinements and (structural) performance analysis are overviewed in Section 3. Section 4 describes the systematic derivation of place invariants by net refinements. Application of the proposed approach to analysis of manufacturing cell schedules is presented in Section 5. Section 6 contains some concluding remarks.

2. Schedules for manufacturing cells

A simple manufacturing cell with three machines, $M1$, $M2$ and $M3$, an input conveyor In, an output conveyor Out, and a robot, is outlined in Fig.2.1.

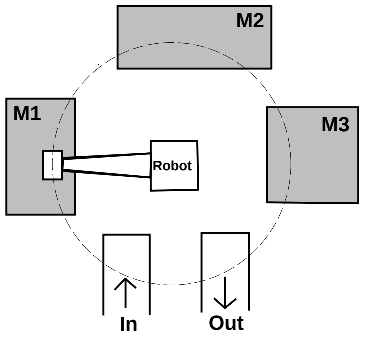


Fig.2.1. Layout of a 3–machine cell.

Sequences of operations executed for consecutive parts (transported to the cell by the input conveyor) are called schedules. These schedules can be described by sequences of 'move' operations (performed by the robot) such as picking a part from the input conveyor, transporting it to a machine and loading it, then (when the machine's operation is finished) unloading, transporting to another machine, and so on. It is known [14] that for a cell with m machines, there are $m!$ different schedules. The optimal schedule is usually the one which maximizes the throughput of the cell (or minimizes its cycle time).

Assuming (for simplicity) that all parts follow the same path through the cell, and that this path is from the input conveyor to the first machine, then from the first machine to the second, and so on, the six 3–machine schedules, denoted here A, B, ..., F, are as follows (in the schedules, element '01' denotes the operation of picking a new part from the input conveyor ('0'), moving it to the first machine ('1') and loading the part; similarly, element '12' denotes unloading the first machine, transporting the unloaded part to the second machine and loading the part, and element '30' – unloading the third machine, moving to the output conveyor and depositing the part there):

A:	01 – 12 – 23 – 30
B:	01 – 12 – 30 – 23
C:	01 – 23 – 12 – 30
D:	01 – 23 – 30 – 12
E:	01 – 30 – 12 – 23
F:	01 – 30 – 23 – 12

Since all schedules are cyclic, it is (arbitrarily) assumed that a uniform 'beginning' of all schedules is the operation of picking a new part from the input, transporting it and loading on the first machine of the cell (step '01').

It has been shown [19] that the schedules for a 3–machine cell can conveniently be derived from the two schedules for a 2–machine cell:

01 – 12 – 20
01 – 20 – 12

by first replacing steps '20' by '23' (which is due to the additional machine in the cell), and then inserting the new step '30' in all three possible positions of each 2–machine schedule, i.e., after the third entry, between the second and the third entries, and between the first and the second entries (the first step remains '01' as the 'standard beginning').

In the same way, the 3–machine schedules can be expanded into 4–machine ones (there are 24 such schedules), and the 2–machine schedules can be derived from the single 1–machine schedule:

01 – 10 .

This single schedule can be used as the 'standard' initial schedule for a hierarchical derivation of schedules for any other cell [19].

Each of these schedules can easily be transformed into a complete sequence of robot's operations [18] by adding all those robot's moves which are necessary to perform the required transport functions. For example, the 3–machine schedule D, '01 – 23 – 30 – 12', requires that the robot, after loading the new part on machine $M1$ (element '01'), moves to machine $M2$ to unload it, transport the unloaded part to machine $M3$ and load the part (step '23'); then the robot waits until the $M3$'s operation is finished, unloads the part, transports is to the output conveyor and drops it there (step '30'), after which it moves to machine $M1$ to unload it and transport the unloaded part to machine $M2$ (step '12'), and finally moves to the input conveyor to start another cycle. Such a detailed schedule of the robot is a sequence of steps in which the robot carries a part from one component of the cell to another (such steps are denoted by $\Rightarrow$), or performs 'empty' moves, without carrying a part (such steps are denoted by $\rightarrow$).

The discussed schedule D is a (cyclic) sequence of the following steps:

$$In \Rightarrow M1 \rightarrow M2 \Rightarrow M3 \Rightarrow Out \rightarrow M1 \Rightarrow M2 \rightarrow In.$$

3. Nets, invariants and refinements

This section recalls basic concepts of timed Petri nets, net invariants and net refinements. A more detailed discussion can be found in [11, 13, 17, 18].

A place/transition (ordinary, i.e., with no arc weights) net $\mathbf{N}$ is a triple $\mathbf{N} = (P, T, A)$ where P is a finite, nonempty set of places, T is a finite, nonempty set of transitions, A is a set of directed arcs, and $A \subseteq P \times T \cup T \times P$, such that for each transition there exists at least one place connected with it. For each place p (and each transition t) the input set, $Inp(p)$ (or $Inp(t)$), is the set of transitions (or places) connected by directed arcs to p (or t). The output sets, $Out(p)$ and $Out(t)$, are defined similarly.

A marked Petri net $\mathbf{M}$ is a pair $\mathbf{M} = (\mathbf{N}, m_0)$ where $\mathbf{N}$ is a Petri net, $\mathbf{N} = (P, T, A)$, and m_0 is an initial marking function, $m_0 : P \rightarrow \{0, 1, ...\}$ which assigns a (nonnegative) integer number of tokens to each place of the net.

Let any function $m : P \rightarrow \{0, 1, ...\}$ be called a marking in a net $\mathbf{N} = (P, T, A)$.

A transition t is enabled by a marking m iff every input place of this transition contains at least one token. Every transition enabled by a marking m can fire. When a transition fires, a token is removed from each of its input places and a token is added to each of its output places. This determines a new marking in a net, a new set of enabled transitions, and so on. The set of all markings that can be derived from the initial marking is called the set of reachable markings. If this set if finite, the net is bounded.

A place p is shared iff it is an input place for more than one transition. A net is (structurally or statically) conflict–free if it does not contain shared places. A marked net is (dynamically) conflict–free if for any marking in the set of reachable markings, and for any shared place, at most one of transitions sharing the place is enabled. Only bounded conflict–free nets are considered in this paper.

Each place/transition net $\mathbf{N} = (P, T, A)$ can conveniently be represented by a connectivity (or incidence) matrix $\mathbf{C} : P \times T \rightarrow \{-1, 0, +1\}$ in which places correspond to rows, transitions to columns, and the entries are defined as:

$$\forall\, p \in P\ \forall\, t \in T : \mathbf{C}[p, t] = \begin{cases} -1, & \text{if } t \in Out(p) - Inp(t), \\ +1, & \text{if } t \in Inp(p) - Out(p), \\ 0, & \text{otherwise.} \end{cases}$$

If a marking m_j is obtained from another marking m_i by firing a transition t_k then (in vector notation) $m_j = m_i + \mathbf{C}[k]$, where $\mathbf{C}[k]$ denotes the k-th column of $\mathbf{C}$, i.e., the column representing t_k.

Connectivity matrices disregard 'selfloops', that is, pairs of arcs (p, t) and (t, p); any firing of a transition t cannot change the marking of p in such a selfloop, so selfloops are neutral with respect to token count of a net. A pure net is defined as a net without selfloops [13].

A P–invariant (place invariant) of a net $\mathbf{N}$ is any nonnegative, nonzero integer (column) vector I which is a solution of the matrix equation

$$\mathbf{C}^T \times I = 0,$$

where $\mathbf{C}^T$ denotes the transpose of matrix $\mathbf{C}$. It follows immediately from this definition that if I_1 and I_2 are P–invariants of $\mathbf{N}$, then also any linear (positive) combination of I_1 and I_2 is a P–invariant of $\mathbf{N}$.

A basic P–invariant of a net is defined as a P–invariant which does not contain simpler invariants. All basic P–invariants I of ordinary nets are binary vectors [13], $I : P \rightarrow \{0, 1\}$.

A net $\mathbf{N}_i = (P_i, T_i, A_i)$ is a P_i-implied subnet of a net $\mathbf{N} = (P, T, A)$, $P_i \subset P$, iff:

(1) $T_i = \{t \in T \mid \exists\, p \in P_i : (p, t) \in A \vee (t, p) \in A\}$,

(2) $A_i = A \cap (P_i \times T \cup T \times P_i)$.

It should be observed that in a (pure) net $\mathbf{N}$, each P–invariant I of $\mathbf{N}$ determines a P_I-implied (invariant) subnet of $\mathbf{N}$, where $P_I = \{p \in P \mid I(p) > 0\}$; P_I is sometimes called the support of the invariant I; all nonzero elements of I select rows of $\mathbf{C}$, and each selected row i corresponds to a place p_i with all its input (+1) and all output (–1) arcs associated with it.

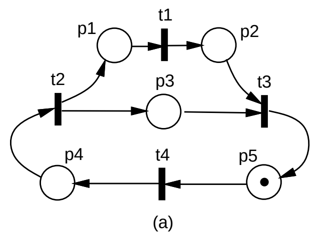
For the Petri net shown in Fig.3.1.a, the connectivity matrix is:

$\mathbf{C}$	t_1	t_2	t_3	t_4
p_1	-1	$+1$	0	0
p_2	$+1$	0	-1	0
p_3	0	$+1$	-1	0
p_4	0	-1	0	$+1$
p_5	0	0	$+1$	-1

and there are two basic invariants, $I_1 = [1, 1, 0, 1, 1]$, i.e., $\{p_1, p_2, p_4, p_5\}$, and $I_2 = [0, 0, 1, 1, 1]$, i.e., $\{p_3, p_4, p_5\}$. It can be observed that the basic invariants correspond to the smallest subsets of rows of the connectivity matrix for which the (component–wise) sums are equal to (vector) zero.

The P_I–implied subnets are simple cycles, (p_1, t_1, p_2, t_3, p_5, t_4, p_4, t_2) and (p_3, t_3, p_5, t_4, p_4, t_2).

Finding basic invariants is a 'classical' problem of linear algebra, and there are known algorithms to solve this problem efficiently [9, 10].



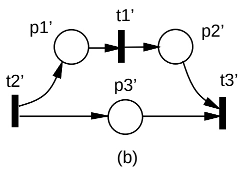


Fig.3.1. Petri net $\mathbf{M}_0$ (a) and its refinement $\mathbf{M}_1$ (b).

Refinements in Petri nets can be defined in several ways; a convenient approach, proposed in [19], refines a net by replacing a single element (a transition or a place) by a subnet connected to the input and output sets of the replaced element.

More formally, a refinement system $\mathcal{R}$ is defined as a 5–tuple, $\mathcal{R} = (\mathbf{M}_0, \mathcal{M}, \rho, \phi, \psi)$, where:

$\mathbf{M}_0$ is a marked (initial) place/transition net, $\mathbf{M}_0 = (P_0, T_0, A_0, m_0)$;

$\mathcal{M}$ is a family of (marked) place/transition refinement nets, $\mathcal{M} = \{\mathbf{M}_1, ..., \mathbf{M}_k\}$;

ρ is a (partial) refinement function which associates elements of P_0 (place refinements) and T_0 (transition refinements) with nets from $\mathcal{M}$, $\rho : P_0 \cup T_0 \rightarrow \{1, ..., k\}$ such that if $p \in Dom(\rho)$ and $t \in Dom(\rho)$, then $(p,t) \notin A_0$ and $(t,p) \notin A_0$; each place $p \in P_0$ is refined by the net $\mathbf{M}_{\rho(p)}$ if $p \in Dom(\rho)$, otherwise p remains a simple place, and each transition $t \in T_0$ is refined by $\mathbf{M}_{\rho(t)}$ if $t \in Dom(\rho)$;

ϕ and ψ are (input and output) interface functions which define the interconnections between the input and output sets of a place (or transition) and its refinement determined by ρ; for each $p \in P_0$, if $p \in Dom(\rho)$, then $\phi(p) : T_0 \rightarrow 2^{P_{\rho(p)}}$ and $\psi(p) : T_0 \rightarrow 2^{P_{\rho(p)}}$ such that $\phi(p)(t)$ is undefined if $(t,p) \notin A_0$ and $\psi(p)(t)$ is undefined if $(p,t) \notin A_0$; similarly, for each $t \in T_0$, if $t \in Dom(\rho)$, then $\phi(t) : P_0 \rightarrow 2^{T_{\rho(t)}}$ and $\psi(t) : P_0 \rightarrow 2^{T_{\rho(t)}}$ such that $\phi(t)(p)$ is undefined if $(p,t) \notin A_0$ and $\psi(t)(p)$ is undefined if $(t,p) \notin A_0$.

It should be noted that a single refinement can replace several (nonadjacent) elements of a net $\mathbf{M}_0$, but only refinements of single elements are used in this paper.

For example, Fig.3.1.a shows a net that can be refined by using the net shown in Fig.3.1.b as a replacement of transition t_3; i.e., first t_3 and all arcs incident with it are removed from the net in Fig.3.1.a, and then the remaining net is connected to the net in Fig.3.1.b by new arcs: from p_2 to t_2', from p_3 to t_2', and from t_3' to p_5. The resulting net is shown in Fig.3.2.

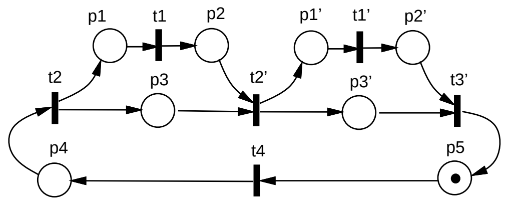


Fig.3.2. Refined net.

For this example, $\mathbf{M}_0$ is the net from Fig.3.1.a, $\mathcal{M} = \{\mathbf{M}_1\}$, where $\mathbf{M}_1$ is the net shown in Fig.3.1.b, and the partial functions ρ, ϕ and ψ are as follows:

$$\forall\ x \in P_0 \cup T_0 : \rho(x) = \begin{cases} 1, \text{if } x = t_3; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\ p \in P_0 : \phi(t_3)(p) = \begin{cases} \{t_2'\}, \text{if } p \in \{p_2, p_3\}; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\ p \in P_0 : \psi(t_3)(p) = \begin{cases} \{t_3'\}, \text{if } p = p_5; \\ \text{undefined otherwise.} \end{cases}$$

It should be observed that a net isomorphic to the one shown in Fig.3.2 can be obtained by refining t_2 in Fig.3.1.a with the same net $\mathbf{M}_1$ shown in Fig.3.1.b. In this case, the partial functions ρ, ϕ and ψ are:

$$\forall\ x \in P_0 \cup T_0 : \rho(x) = \begin{cases} 1, \text{if } x = t_2; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\ p \in P_0 : \phi(t_2)(p) = \begin{cases} \{t_2'\}, \text{if } p = p_4; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\ p \in P_0 : \psi(t_2)(p) = \begin{cases} \{t_3'\}, \text{if } p \in \{p_1, p_3\}; \\ \text{undefined otherwise.} \end{cases}$$

In timed Petri nets each transition takes a 'real time' to fire, i.e., there is a 'firing time' associated with each transition of a net which determines the duration of transition's firings.

A conflict–free timed Petri net $\mathbf{T}$ is a pair $\mathbf{T} = (\mathbf{M}, f)$ where:

$\mathbf{M}$ is a conflict–free marked Petri net, $\mathbf{M} = (\mathbf{N}, m_0)$, $\mathbf{N} = (P, T, A)$,

f is a firing time function which assigns the nonnegative (average) firing time $f(t)$ to each transition t of the net, $f : T \rightarrow \mathbf{R}^{\oplus}$, and $\mathbf{R}^{\oplus}$ denotes the set of nonnegative real numbers.

The behavior of a timed Petri net can be represented by a sequence of 'states' and state transitions where each 'state' describes the distribution of tokens in places as well as firing transitions of the net; detailed definitions of states and state transitions are given in [17]. The states and state transitions can be combined into a graph of reachable states; this graph is a semi–Markov process defined by the timed net **T**. For cyclic conflict–free timed nets, such state graphs are simple cycles which represent the cyclic behavior of such nets. Each such timed Petri net contains a basic invariant subnet with the cycle time equal to the cycle time of the whole net. All other subnets, with smaller cycle times, will be subjected to some synchronization delays, imposed by the 'slowest' subnet that determines the cycle time of the whole net. The cycle time of the net is thus equal to the maximum cycle time if its basic invariant subnets.

For example, the net shown in Fig.3.2 has four basic P–invariants and four cyclic subnets implied by these P–invariants; these subnets contain the following transitions:

subnet	*transitions*
1	t_2, t'_2, t'_3, t_4
2	$t_2, t_1, t'_2, t'_3, t_4$
3	$t_2, t'_2, t'_1, t'_3, t_4$
4	$t_2, t_1, t'_2, t'_1, t'_3, t_4$

The cycle time τ_0 of this net is determined by the last subnet as the sets of transitions of all other subnets are subsets of the last one's (so the cycle times of these other subnets are smaller than the last one's):

$$\tau_0 \;=\; f(t_1) + f(t'_1) + f(t_2) + f(t'_2) + f(t'_3) + f(t_4).$$

4. Refinements of place invariants

It can be observed that a transition refinement is a transformation of the connectivity matrix in which a single column (the refined transition) is replaced by a set of new columns (the transitions of the refining net) and possibly a set of new rows (places of the refining net). If the refined transition is represented by the last colum of the connectivity matrix, the structure of the connectivity matrix after refinement is

A'	
B'	**B"**
	A"

where **A'** is the section of the original connectivity matrix which is not related to the refined transition, **B'** and **B"** are the rows corresponding to places connecting the 'old' net with the 'new' one, and **A"** is the part of the refining net that is not affected by the refinement. The two 'blank' submatrices contain only zeroes.

The basic P–invariants of the refined matrix can be divided into three types:

- 'old' P–invariants which are entirely within the original net and are not affected by the refinement; they correspond to such subsets of rows of the original connectivity matrix which, in the column of the refined transition, have only zero entries; all these invariants are invariants of the submatrix **A'**;
- 'new' P–invariants which are entirely within the refining net and are 'brought into' the refined net by the refinement; they correspond to subsets of rows of the new submatrix **A"** of the connectivity matrix;
- 'mixed' P-invariants which, in the original net, imply the refined transition, so, after the refinement, they include 'new' and 'old' net elements; all such invariants can be obtained by 'combinations' of the original P–invariants with the invariants of the refining net; submatrices **B'** and **B"** are involved in all 'mixed' invariants.

The total number of P–invariants of a refined net is thus equal to the sum $N_{old} + N_{new} + N_{mixed}$.

The P–invariants of the 'mixed' type can be obtained by combining those P–invariants of the original net that imply the refined transition, with P–invariants of the augmented refining net; the augmented net contains one or more additional places (for transition refinements, or one or more additional transitions for place refinements) that represent the connections indicated by the interface input and output functions ϕ and ψ.

If the original net contains N_r P–invariants that imply the refined transition t, and if the augmented refining net contains N_a P–invariants which include the augmenting place(s), the number of the mixed invariants in the refined net is:

$$N_{mixed} \;=\; N_r * N_a.$$

For the example discussed in the previous section (for convenience, the columns corresponding to t_3 and t_4 are swapped so the refined transition, t_3, is the rightmost column of the connectivity matrix), the connectivity matrix for the refined net (Fig.3.2) is as follows:

	t_1	t_2	t_4	t'_1	t'_2	t'_3
p_1	-1	$+1$	0	0	0	0
p_4	0	-1	$+1$	0	0	0
p_2	$+1$	0	0	0	-1	0
p_3	0	$+1$	0	0	-1	0
p_5	0	0	-1	0	0	$+1$
p'_1	0	0	0	-1	$+1$	0
p'_2	0	0	0	$+1$	0	-1
p'_3	0	0	0	0	$+1$	-1

with separation of the matrix into the six submatrices discussed earlier.

All invariants of this connectivity matrix are of the 'mixed' type (i.e., $N_{old} = N_{new} = 0$).

Fig.4.1 shows the augmented refining net $\mathbf{M}_1$ (Fig.3.1.b); the additional place p_0 is an input place for t'_2 because the input interface function ϕ is defined only for t'_2, and it is an output place for t'_3 because the output interface function ψ is defined only for t'_3, and there is, in the original net $\mathbf{M}_0$, a directed path from p_5 (the connection created by ψ) to p_2 as well as to p_3 (the connections created by ϕ).

This augmented net has two P–invariants, one with the subset of places $\{p_0, p'_3\}$, and the other with $\{p_0, p'_1, p'_2\}$.

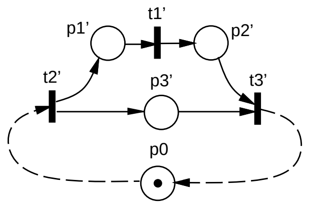


Fig.4.1. Augmented refining net $\mathbf{M}_1$.

Since the original net (Fig.3.1.a) has two P–invariants which imply transition t_3, there are four P–invariants for the refined net, obtained by combining each of the two original P–invariants with each of the two refining ones:

invariant	*places*
1	p_1, p_2, p_4, p_5, p'_3
2	$p_1, p_2, p_4, p_5, p'_1, p'_2$
3	p_3, p_4, p_5, p'_3
4	$p_3, p_4, p_5, p'_1, p'_2$

It can be easily verified that the subnets implied by these P–invariants contain the following sets of transitions:

subnet	*transitions*
1	$t_2, t_1, t'_2, t'_3, t_4$
2	$t_2, t_1, t'_2, t'_1, t'_3, t_4$
3	t_2, t'_2, t'_3, t_4
4	$t_2, t'_2, t'_1, t'_3, t_4$

so the cycle time of this model is determined by the subnet implied by P–invariant (2) (the sets of transitions of all other implied subnets are subsets of (2)).

Fig.4.2 shows a Petri net model of schedule D for a 3–machine cell, as described in Section 2. The three machines (or rather machine operations) are represented by transitions t_1, t_2 and t_3, each with its input place (to represent the condition 'part is loaded') and an output place (to indicate that the 'machine's operation is finished'). The remaining part of the model is a cyclic representation of the sequence of robot's steps.

The initial marking function represents the configuration when a 'finished' part has been dropped on the output conveyor.

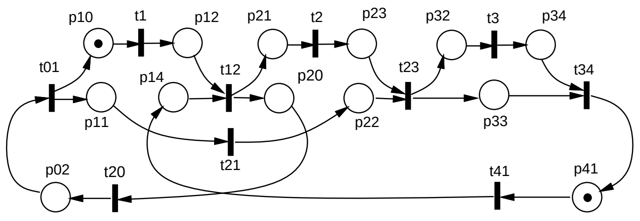


Fig.4.2. Model of schedule D.

The interpretation of robot's steps is as follows (the names of transitions are consistent with [18]):

trans.	*robot operations*
t_{01}	pick a part from In, move to $M1$ and load
t_{12}	unload $M1$, move to $M2$ and load
t_{20}	move from $M2$ to In
t_{21}	move from $M1$ to $M2$
t_{23}	unload $M2$, move to $M3$ and load
t_{34}	unload $M3$, move to Out and drop
t_{41}	move from Out to $M1$

The net shown in Fig.4.2 can be obtained from that in Fig.3.2 by a refinement of transition t_2 using the refining net M_2 shown in Fig.4.3.a (it can also be derived in a different way).

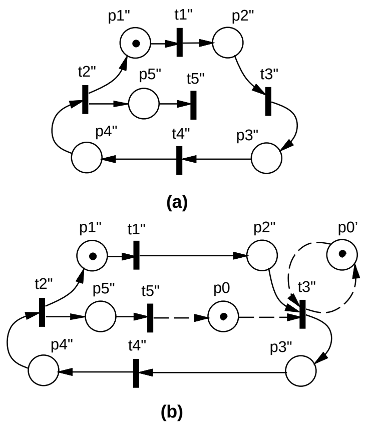


Fig.4.3. Refining net $\mathbf{M}_2$ (a) and its augmented version (b).

In this case, the (partial) refining functions ρ, ϕ and ψ are (assuming that $\mathcal{M} = \{\mathbf{M}_2\}$):

$$\forall\, x \in P_0 \cup T_0 : \rho(x) = \begin{cases} 2, \text{if } x = t_2; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\, p \in P_0 : \phi(t_2)(p) = \begin{cases} \{t''_3\}, \text{if } p = p_4; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\, p \in P_0 : \psi(t_2)(p) = \begin{cases} \{t_3''\}, \text{if } p = p_1; \\ \{t_5''\}, \text{if } p = p_3; \\ \text{undefined otherwise.} \end{cases}$$

The augmented refining net contains one 'mixed' P–invariant with places p_0, p_3'', p_4'' and p_5'', and one 'local' P–invariant (with places p_1'', p_2'', p_3'', p_4''). The net from Fig.3.2 has four P–invariants, all of which imply t_2; the total number of P–invariants for the net shown in Fig.4.2 is thus 5. The self–loop in the augmented net (in Fig.4.3.b) indicates that, after the refinement, some 'mixed' P–invariants will actually contain only the elements of the 'old' net and interfacing arcs (created by functions ϕ and ψ) but not any elements of the refining net. The five P–invariants are:

invariant	*places*
1	$p_1'', p_2'', p_3'', p_4''$
2	p_1, p_2, p_3', p_5, p_4
3	$p_1, p_2, p_1', p_2', p_5, p_4$
4	$p_3'', p_4'', p_5'', p_3, p_3', p_5, p_4$
5	$p_3'', p_4'', p_5'', p_1, p_2, p_3', p_5, p_4$

The same invariants are determined directly from the net model in [18].

5. Evaluation of models

The determination of the cycle time τ_0 of the model (or the throughput of the cell) can be based on P–invariant–implied subnets. More specifically, for a timed net model $\mathbf{T} = (\mathbf{M}, f)$ covered by a family $\mathcal{I}$ of conflict–free P–invariant–implied subnets, the cycle time is:

$$\tau_0 \;=\; \max_{P_{I_i} \in \mathcal{I}} (\tau_i)$$

where the cycle time τ_i of the P_{I_i}–implied conflict–free subnet $\mathbf{M}_i = (\mathbf{N}_i, m_i)$, $\mathbf{N}_i = (P_i, T_i, A_i)$, is [5]:

$$\tau_i \;=\; \frac{\sum_{t \in T_i} f(t)}{\sum_{p \in P_i} m_i(p)}.$$

For the model shown in Fig.4.2, the firing times of transitions t_1, t_2 and t_3 characterize the (average) operation times of machines $M1$, $M2$ and $M3$, respectively; the firing times of all remaining transitions can be determined on the basis of actions performed by the robot:

transition	*execution time*
t_{01}	$u + w + y$
t_{12}	$v + w + y$
t_{20}	$2y$
t_{21}	y
t_{23}	$v + w + y$
t_{34}	$v + x + y$
t_{41}	$2y$

where the 'execution times' are expressed in terms of more elementary operations:

u – the (average) pickup time,
v – the (average) unload time,
w – the (average) load time,
x – the (average) drop time and
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 $M3$ to Out, Out to In and In to $M1$ moves).

The P–invariant–implied subnets contain the following sets of transitions:

subnet	*transitions*
1	$t_1, t_{12}, t_{20}, t_{01}$
2	$t_2, t_{23}, t_{34}, t_{41}, t_{12}$
3	$t_2, t_{23}, t_3, t_{34}, t_{41}, t_{12}$
4	$t_{01}, t_{21}, t_{23}, t_{34}, t_{41}, t_{12}, t_{20}$
5	$t_{01}, t_{21}, t_{23}, t_3, t_{34}, t_{41}, t_{12}, t_{20}$

Since the set of transitions of subnet (2) is a subset of (3), and (4) is a subset of (5), the cycle time is:

$$\tau_0 \;=\; \max(\tau_1, \tau_3, \tau_5)$$

where:

$$\begin{aligned} \tau_1 &= o_1 + u + v + 2w + 4y, \\ \tau_3 &= o_2 + o_3 + 3v + 2w + x + 5y, \\ \tau_5 &= o_3 + u + 3v + 3w + x + 9y, \end{aligned}$$

and $o_1 = f(t_1)$, $o_2 = f(t_2)$, $o_3 = f(t_3)$.

The same approach can be used for deriving schedules for 4–machine cells from schedules for 3–machine cells. Fig.5.1 shows a model of one of 24 schedules for a 4–machine cell:

$$In \Rightarrow M1 \rightarrow M3 \Rightarrow M4 \Rightarrow Out \rightarrow M2 \Rightarrow M3 \rightarrow M1 \Rightarrow M2 \rightarrow In.$$

and Tab.5.1 – the operations represented by transitions and their execution times (using the same convention as before).

This model can be obtained from the net shown in Fig.4.2 by introducing a new, additional machine as a refinement of transition t_{12}; this new machine becomes machine $M2$; the previous machine $M2$ is renamed as the new $M3$, and the previous $M3$ becomes the new $M4$.

The refining net $\mathbf{M}_3$ is shown in Fig.5.2.a, and the refining functions are:

$$\forall\, x \in P_0 \cup T_0 : \rho(x) = \begin{cases} 3, \text{if } x = t_{12}; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\, p \in P_0 : \phi(t_{12})(p) = \begin{cases} \{t_2'\}, \text{if } p = p_{12}; \\ \{t_3'\}, \text{if } p = p_{14}; \\ \text{undefined otherwise;} \end{cases}$$

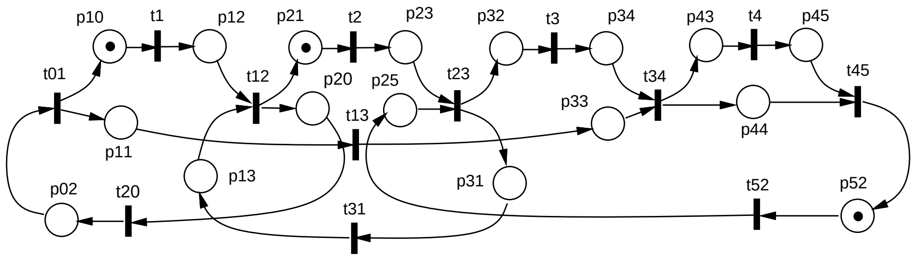


Fig.5.1. Net model of a 4–machine cell's schedule.

transition	*robot operations*	*execution times*
t_{01}	pick a part from In, move to $M1$ and load	$u + w + y$
t_{12}	unload $M1$, move to $M2$ and load	$v + w + y$
t_{13}	move from $M1$ to $M3$	$2y$
t_{20}	move from $M2$ to In	$2y$
t_{23}	unload $M2$, move to $M3$ and load	$v + w + y$
t_{31}	move from $M3$ to $M1$	$2y$
t_{34}	unload $M3$, move to $M4$ and load	$v + w + y$
t_{45}	unload $M4$, move to Out and drop	$v + x + y$
t_{52}	move from Out to $M2$	$3y$

Tab.5.1. Transitions, robot operations and their execution times.

$$\forall\ p \in P_0 : \psi(t_{12})(p) = \begin{cases} \{t'_2\}, \text{if } p = p_{20}; \\ \{t'_3\}, \text{if } p = p_{32}; \\ \text{undefined otherwise.} \end{cases}$$

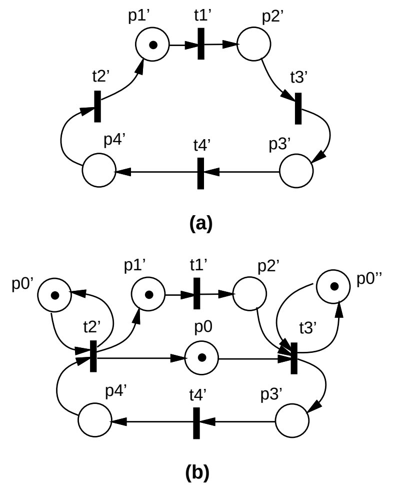


Fig.5.2. Refining net $\mathbf{M_3}$ (a) and its augmented version (b).

The augmented refining net (Fig.5.2.b) contains one 'mixed' P–invariant (with places p_0, p'_3, and p'_4), and one 'local' P–invariant (with places p'_1, p'_2, p'_3, p'_4). Since all five P–invariants of the net in Fig.4.2 imply t_{12}, the total number of P–invariants for the refined net is 6, and the implied subsets of transitions are:

subnet	*transitions*
1	$t_2, t_{23}, t_{31}, t_{12}$
2	$t_1, t_{12}, t_{20}, t_{01}$
3	$t_3, t_{34}, t_{45}, t_{52}, t_{23}$
4	$t_3, t_{34}, t_4, t_{45}, t_{52}, t_{23}$
5	$t_{01}, t_{13}, t_{34}, t_{45}, t_{52}, t_{23}, t_{31}, t_{12}, t_{20}$
6	$t_{01}, t_{13}, t_{34}, t_4, t_{45}, t_{52}, t_{23}, t_{31}, t_{12}, t_{20}$

Since the set (3) is a subset of (4), and (5) is a subset of (6), the cycle time, τ_0, for this model is:

$$\tau_0 \ = \ \max(\tau_1, \tau_2, \tau_4, \tau_6)$$

where:

$$\begin{aligned} \tau_1 &= o_2 + 2v + 2w + 4y, \\ \tau_2 &= o_1 + u + v + 2w + 4y, \\ \tau_4 &= o_3 + o_4 + 3v + 2w + x + 6y, \\ \tau_6 &= o_4 + u + 4v + 4w + x + 14y. \end{aligned}$$

Another schedule for a 4–machine cell:

$$In \Rightarrow M1 \rightarrow M3 \Rightarrow M4 \Rightarrow Out \rightarrow M1 \Rightarrow M2 \Rightarrow \\ M3 \rightarrow In$$

also derived from the model of a 3–machine schedule of Fig.4.2, is shown in Fig.5.3. In this case the transition t_{12} is refined by the net shown in Fig.3.1.b.

It is shown in [19] that, using transition refinements, only a few refining nets are needed to derived the complete set of schedules for manufacturing cells with any numbers of machines.

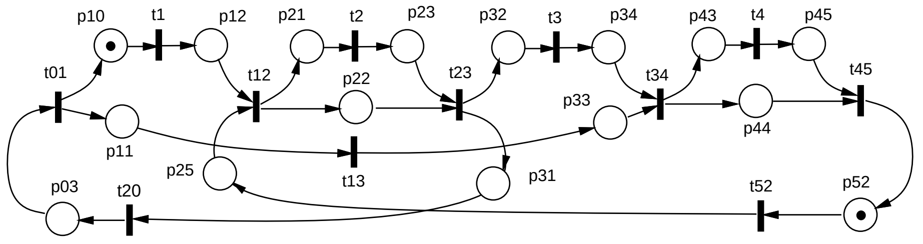


Fig.5.3. Net model of another 4–machine cell's schedule.

6. Concluding remarks

An approach to systematic derivation and evaluation of schedules for manufacturing cells is presented in this paper. The approach uses timed Petri net models and is based on transition refinements which introduce the components of the cell one after another, gradually increasing the complexity of the schedules.

The approach is based on formally defined transformations of net models, in which single transitions are replaced by subnets that refine the model by providing more detailed representations of the modeled activities. It appears that the same refinements which are used for model development, can also be used for systematic derivations of the sets of basic places invariants for the derived models. Since the models of schedules for manufacturing cells are conflict–free nets covered by their place invariant implied subnets, the derived place invariants can be used very conveniently for the performance characterization of the derived models.

Stepwise refinements of transitions must be performed in a 'consistent' way in order to preserve the basic properties of models (such as liveness, boundedness, etc.); to be consistent, the refining nets must satisfy certain conditions (these conditions and the selection of refining nets are not discussed in this paper). For example, all 'mixed' P–invariants must be marked, as must be all 'new' P–invariants of the refining nets (these requirements can easily be checked in the previous examples).

The results of stepwise refinements of transitions can be organized in a hierarchical structure, in which more complicated models (and their place invariants) are systematically derived from simpler ones. Quite often a more complicated model can be derived in more than one way from simpler models. It should be observed that some of the discussed refinements can actually be simplified by using several refinement steps instead of a single refinement; for example, the refining net shown in Fig.4.3.a could be replaced by a simpler net shown in Fig.5.2.a and an additional refinement of place p_3 in Fig.3.2. All such equivalent refinements can be captured by the hierarchical structure of model derivations.

The stepwise refinements of place invariants can be used for reducing the number of cases that need to be considered for finding the best schedule. Since the cycle time of the net is determined by the maximum cycle time of subnets implied by place invariants, in the mixed invariants (Section 4) only the 'maximum' components (i.e., the components with the maximum total firing times of transitions) need to be taken into account. Consequently, instead of $N_{mixed} = N_r * N_a$ mixed P–invariants, only the maximum mixed P–invariant is needed. Similarly, only one 'old' and one 'new' P–invariant needs to be taken into account in each refinement.

The composition of place invariants can also be used to reduce the number of schedules that need to be considered in finding the optimal schedule; the optimal schedule in this case is the schedule with the shortest cycle time. If it can be predicted that the cycle time of a schedule obtained by transition refinement cannot be shorter than the cycle time of some other schedule, there is no need to analyze this refinement any further. For example, quite often the same net can be used as a refinement of several transitions, generating models of several different more complex schedules (the net shown in Fig.5.2.a can be used to refine transition t_{12} in Fig.4.2, which results in the model shown in Fig.5.3, but it can also be used to refine transition t_{23} in Fig.4.2, or transition t_{34} there, creating yet other schedules for a 4–machine cell). If a schedule with the shortest cycle time is sought, only one of these several possible refinements, the one with the shortest cycle time, needs to be analyzed further.

Since the results of invariant analysis are obtained in analytical form, the performance of specific models can easily be obtained by evaluating the derived formulas for the sets of model–specific parameters.

Many simplifying assumptions were made in this paper in order to make the presentation straightforward to follow. These simplifications can easily be removed by appropriate generalizations of the model and its interpretation. For example, more accurate representations of temporal aspects of robot actions may result in a more complicated description of the firing times of transitions (or a more complicated model), but otherwise will have no significant effect on the proposed approach.

Similarly, if a manufacturing cell uses several identical machines instead of just one machine, the model needs only a minor modification to explicitly represent the number of available machines by an additional place with the corresponding initial marking. It should be observed that this additional place will increase the number of P–invariants (and their subnets), but otherwise will not affect the presented approach.

Finally, it is expected that a very similar approach can be used for derivation and analysis of composite schedules for manufacturing cells, i.e., schedules in which more than one part enters the cell and more than one part leaves the cell in each cycle (although the parts that enter the cell may not be the same as that that leave the cell). Composite schedules can be used for manufacturing cells which process several different parts 'simultaneously'. It has been shown [18] that composite schedules can be decomposed into combinations of simple schedules, in which one simple schedule is embedded in another. Net models of composite schedules are (statically) free-choice nets but dynamically conflict–free; their analysis requires an extension of the approach described in this paper.

Acknowledgements

Remarks of four anonymous referees were helpful in revising and improving this paper.

The Natural Sciences and Engineering Research Council of Canada partially supported this research through Research Grant OGP-8222.

References

[1] Banaszak, Z., "Modeling of manufacturing systems"; in: *Modern Manufacturing*, pp.253–286, Springer–Verlag 1994.

[2] Carlier, J., Chretienne, P., Girault, C., "Modelling scheduling problems with timed Petri nets"; in: *Advances in Petri Nets 1984* (Lecture Notes in Computer Science 188), pp.62-82, Springer–Verlag 1985.

[3] Cavalieri, S., Mirabella, O., Zingarino, G., "A Petri net based approach for FMS performance evaluation"; Proc. 23-rd Int. Conf. on Industrial Electronics, Control, and Instrumentation (IECON'97), New Orleans, LA, vol.3, pp.1204–1209, 1997.

[4] Desrochers, A.A., Al-Jaar, R.Y., *Applications of Petri nets in manufacturing systems*; IEEE Press 1995.

[5] DiCesare, F., Harhalakis, G., Proth, J.M., Silva, M., Vernadat, F.B., *Practice of Petri nets in manufacturing*; Chapman & Hall 1993.

[6] Ezpeleta, J., Garcia-Valles, F., Colom, J.M., "A class of well structured Petri nets for flexible manufacturing systems"; in: *Application and Theory of Petri Nets 1998* (Lecture Notes in Computer Science 1420), pp.64–83, 1998.

[7] Frey, G., Mossig, K., Schnabel, M., "Assembly line sequencing based on Petri net T-invariants"; Proc. 9-th IFAC Symp. on Information Control in Manufacturing (INCOM'98), Nancy-Metz, France, vol.2, pp.33–38, 1998.

[8] Holliday, M.A., "Deterministic time and analytical models of parallel architectures"; Ph.D. Thesis, Computer Science Department, University of Wisconsin - Madison, Technical Report #652, 1986.

[9] Krueckeberg, F., Jaxy, M., "Mathematical methods for calculating invariants in Petri nets"; in: *Advances in Petri Nets 1987* (Lecture Notes in Computer Science 266), G. Rozenberg (ed.), pp.104-131, Springer–Verlag 1987.

[10] Martinez, J., Silva, M., "Simple and fast algorithm to obtain all invariants of a generalized Petri net"; in: *Applications and Theory of Petri Nets* (Informatik Fachberichte 52); pp.301–310, Springer–Verlag 1982.

[11] Murata, T., "Petri nets: properties, analysis and applications"; *Proceedings of IEEE*, vol.77, no.4, pp.541–580, 1989.

[12] Ramchandani, C., "Analysis of asynchronous concurrent systems by timed Petri nets"; Project MAC Technical Report MAC–TR–120, Massachusetts Institute of Technology, Cambridge MA, 1974.

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

[14] 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.

[15] 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.

[16] Xue, Y., Kieckhafer, R.M., Choobineh, F.F., "Automated construction of GSPN models for flexible manufacturing systems", *Computers in Industry*, vol.37, no.1, pp.17–25, 1998.

[17] 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.

[18] 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, NF, Canada A1B 3X5, 1995 (available through anonymous `ftp` at `ftp.cs.mun.ca/pub/techreports/tr-9503.ps.Z`).

[19] Zuberek, W.M., "Hierarchical derivation of schedules for manufacturing cells"; Proc. 9-th Symp. on Information Control in Manufacturing (INCOM'98), Nancy-Metz, France, pp.423-428, 1998.