IEEE Int. Conference on Systems, Man, and Cybernetics, Tokyo, Japan, 12-15 October 1999, pp.775-780.

Hierarchical Derivation of Petri Net Models of Composite Schedules for Manufacturing Cells

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

Abstract

Composite schedules for manufacturing cells (or robotic cells) are schedules in which several parts enter and leave the cell in each cycle (although the parts which leave the cell are not necessarily the same as the ones that enter the cell). It appears that composite schedules can easily be transformed into timed Petri net models in which the timed transitions represent the actions (including the durations) of the robot and the machines of the cell. Moreover, such models can be derived using stepwise refinements of net models. Hierarchical modeling is obtained by systematic application of the refinement steps.

1. Introduction

Quite often manufacturing systems are composed of clusters of machines arranged in manufacturing cells (or robotic cells), connected by a common transportation system which moves the machined or assembled parts between the cells or between the cells and storage areas [1].

Each manufacturing cell is composed of a number of machines and a robot (or a robotic arm) which moves the parts from one machine to another, from the input conveyor to the first machine and from the last machine to the output conveyor. The (cyclic) sequence of operations performed by the robot is called a schedule. The throughput of a cell depends on the sequence of robot moves as well as on the sequence in which parts enter the cell [6]. Maximizing the throughput of a robotic cell is thus equivalent to finding a schedule with the minimal cycle time.

The behavior of a manufacturing cell can be represented by 'events' and 'activities'; an activity corresponds to an operation performed by a machine or a robot, and an event corresponds to any change of the cell's activities. Different sets of activities determine the 'states' of the system, and in each state, several activities can occur concurrently, for example, an operation can be performed by one of the machines, and the robot can also transport a part from one machine to another. Petri nets provide a simple and convenient formalism for modeling system that exhibit concurrent activities [8, 7]; they have been successfully used in modeling and analysis of manufacturing systems [2, 3, 5, 4].

Place/transition Petri nets are composed of two types of elements, called 'places' and 'transitions'; places represent conditions (in a very general sense) and transitions – events. If all 'input conditions' of an event are satisfied, the event can occur. The dynamic behavior of a net is represented by 'tokens' which are assigned to places, and which represent conditions that are satisfied ('marked places'). These tokens – when an event occurs – change their assignments. The new distribution of tokens creates a new set of satisfied conditions, new events that can occur, and so on.

In timed Petri nets, the durations of modeled activities are also taken into account, and then the behavior of a net is represented by an embedded Markov chain which can be used for performance analysis of the model. For special classes of nets, structural properties can be used for a simple evaluation of basic performance characteristics. It appears that net models of schedules for manufacturing cells belong to this special class of nets; their its performance can be evaluated on the basis of place invariants [12, 13].

In composite schedules, several different parts can be assembled or machined during one (more complex) cycle of operations. The paper shows how simple schedules can be combined into composite ones, by using refinement operations; this refinement is used for a hierarchical development of schedules. The refinement operation used in this paper is a generalization of simple transition refinements used previously [13, 14]; it replaces a collection of net elements in one refinement step. Consequently, an operation of 'fusion' of places or transitions becomes a special case of this refinement. Many other net transformations can also be specified by these generalized net refinements.

Section 2 briefly introduces descriptions of manufacturing cells and their schedules. Section 3 recalls basic concepts of Petri nets and defines the refinement operation used in this paper, while Section 4 applies this refinement to systematic derivation of composite schedules. Section 5 concludes the paper.

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.

It is known [9] that for a cell with m machines, there are $m!$ different schedules. These schedules can be described by sequences of 'move' operations (performed

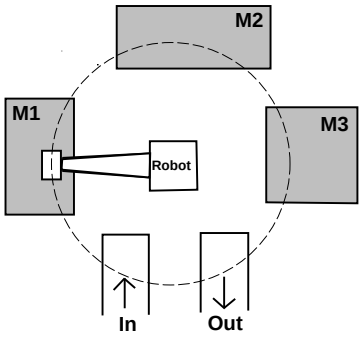

Fig.2.1. Layout of a 3–machine cell.

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 the part, transporting it to another machine, and so on.

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, can be represented by sequences of cell "states". Each state describes the distribution of parts over the machines of the cell, so, for an m–machine cell, it can be represented by a sequence of m (binary) digits, each digit describing one machine of the cell (with the value "0" if the machine is 'empty', and "1" if a part is loaded on the machine; so, '010' is the state in which $M1$ and $M3$ are empty and there is a part loaded on $M2$):

A: 000 – 100 – 010 – 001 – 000
B: 001 – 101 – 011 – 010 – 001
C: 001 – 101 – 100 – 010 – 001
D: 010 – 110 – 101 – 011 – 010
E: 010 – 110 – 101 – 100 – 010
F: 011 – 111 – 110 – 101 – 011

It is (arbitrarily) assumed that all schedules begin in a uniform way by the operation of picking a new part from the input, transporting it and loading on the first machine of the cell (this is why for all schedules the first component of the first state description, i.e., the machine $M1$, is always "0").

Each of these schedules can easily be transformed into a complete sequence of robot's operations [12] by adding all those robot's moves which are necessary to perform the required state transitions. For example, the schedule D, "010 – 110 – 101 – 011 – 010", requires that the robot, after loading the new part on machine $M1$ (state transition 010 – 110), moves to machine $M2$ to unload it, transport the unloaded part to machine $M3$ and load the part (state transition 110 – 101); then the robot moves back to $M1$, unloads it, transports the unloaded part to $M2$ and loads it (transition 101 – 011); next the robot moves to $M3$, unloads the part, transports is to the output conveyor and drops it there (state transition 011 – 010), after which the robot moves to the input conveyor to start another cycle. Such a detailed schedule 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:

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

while for schedule A the sequence is simply:

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

These sequences can be further refined by explicitly showing all operations performed by the robot. Let $P0$ and $D0$ denote the operations of picking a new part from the input conveyor and dropping a part on the output conveyor, respectively; let also Li and Ui denote the operations of loading a part on machine Mi and unloading the part from Mi; then the detailed sequence for schedule D is:

$$P0 \Rightarrow M1, L1 \rightarrow M2, U2 \Rightarrow M3, L3 \rightarrow M1, U1 \Rightarrow M2, L2 \rightarrow M3, U3 \Rightarrow Out, D0 \rightarrow In$$

and the sequence for schedule A is:

$$P0 \Rightarrow M1, L1, U1 \Rightarrow M2, L2, U2 \Rightarrow M3, L3, U3 \Rightarrow Out, D0 \rightarrow In.$$

It appears that composite schedules, in which several parts enter the cell (and several parts leave the cell) in one cycle, are extension of the same approach; for composite schedules, the robot executes operations of several simple schedules. Not surprisingly, composite schedules can be represented by embedding simple schedules one into another in a way that preserves the consistency of state descriptions [12]. For example, it can be observed that the initial state of schedule D (010) is also one of the states in schedule A, so it is possible, in state 010 of schedule A, to "switch" to schedule D, execute it and return to continuation of schedule A, as shown in the following outline:

schedule A		*schedule D*
000		
100		
010	$\longrightarrow$	010
		110
		101
		011
010	$\longleftarrow$	010
001		
000		

A complete sequence of robot operations is a combination of the sequence for schedule A with that for schedule D, with (possibly) some additional moves at the transition from one schedule to another. For the composite schedule A+D this detailed sequence is:

$$P0 \Rightarrow M1, L1, U1 \Rightarrow M2, L2$$
$$\rightarrow In, P0 \Rightarrow M1, L1 \rightarrow M2, U2$$
$$\Rightarrow M3, L3 \rightarrow M1, U1 \Rightarrow M2, L2$$
$$\rightarrow M3, U3 \Rightarrow Out, D0 \rightarrow M2,$$
$$U2 \Rightarrow M3, L3, U3 \Rightarrow Out, D0 \rightarrow In.$$

Although it is possible to derive composite schedules from the detailed state descriptions [12], there is a more intuitively appealing approach which derives models of composite schedules by combining models of simple schedules.

3. Petri nets and net refinements

This section recalls basic concepts of timed Petri nets and net refinements. A more detailed discussion can be found in [7, 8, 11, 12].

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. The Inp and Out notation is extended in the natural way to sets of places and transitions.

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 free–choice if the input sets of all transitions sharing the same place are identical. 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 this place is enabled. Only bounded conflict–free nets are considered in this paper.

Refinements in Petri nets can be defined in several ways; a convenient approach, proposed in [13], 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. A more general refinement operations are used in this paper in order to refine a number of elements in a single step. Consequently, more general net transformations can be described by this refinement operation, with the previously used refinements of places and transitions [13, 14] as special cases.

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, $\mathbf{M}_0 = (\mathbf{N}_0, m_0)$, $\mathbf{N}_0 = (P_0, T_0, A_0)$;

$\mathcal{M}$ 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 subsets of elements of $P_0 \cup T_0$ with nets from $\mathcal{M}$, $\rho : 2^{P_0 \cup T_0} \rightarrow \{1, ..., k\}$ such that:

$$\forall X_i \in Dom(\rho)\ \forall X_j \in Dom(\rho) - \{X_i\} :$$
$$X_i \cap X_j = \emptyset\ \wedge\ X_i \cap (Inp(X_j) \cup Out(X_j)) \cup$$
$$X_j \cap (Inp(X_i) \cup Out(X_i)) = \emptyset,$$

i.e., if $X \in Dom(\rho)$, than all elements of X are replaced by the same net $\mathbf{M}_{\rho(X)}$, and different subsets $X_i, X_j \in Dom(\rho)$ are disjoint and do not contain adjacent elements of $\mathbf{M}_0$;

ϕ and ψ are (input and output) interface functions which define the interconnections between the input and output sets of the refined elements and their refinement nets determined by ρ; for each $X \in Dom(\rho)$, $\phi(X) : T_0 \rightarrow 2^{P_{\rho(p)}} \cup P_0 \rightarrow 2^{T_{\rho(t)}}$ such that $\phi(X)(t)$ is undefined if $t \notin Inp(X)$ and $\phi(X)(p)$ is undefined if $p \notin Inp(X)$; similarly, for each $X \in Dom(\rho)$, $\psi(p) : T_0 \rightarrow 2^{P_{\rho(p)}} \cup P_0 \rightarrow 2^{T_{\rho(t)}}$ such that $\psi(X)(t)$ is undefined if $t \notin Out(X)$ and $\psi(X)(p)$ is undefined if $p \notin Out(X)$.

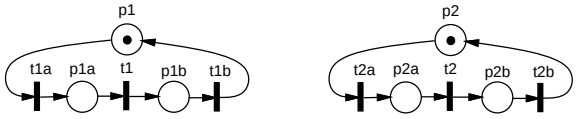

Fig.3.1. Net models of two machines, $\mathbf{M}_1$ and $\mathbf{M}_2$.

For example, Fig.3.1 shows two simple cyclic nets which model two machines of a (very simple) cell. Places p_1 and p_2 indicate (if marked) that the corresponding machine is available for operation (that it is "free"); for each machine Mi, $i = 1, 2$, transition t_i represents the machine's operation, transition t_{ia} the loading of a part on the machine Mi, and transition t_{ib} – unloading the part, after which the machine becomes available for another operation.

Fig.3.2 shows another cyclic net which represents the sequence of robot moves for one of the schedules for a two–machine cell; this sequence is:

$$P0 \Rightarrow M1, L1, U1 \Rightarrow M2, L2, U2 \Rightarrow Out, D0 \rightarrow In.$$

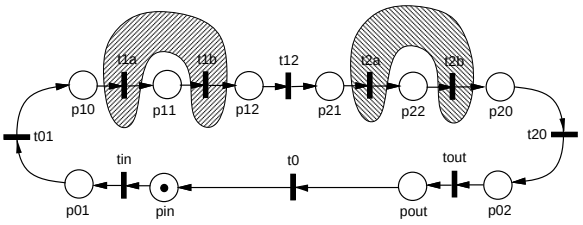

Fig.3.2. Net model of the robot, $\mathbf{M}_0$.

The transitions represent consecutive steps of the sequence, so t_0 represents the move from Out to In, t_{in} – picking a new part from the input conveyor, t_{01} – move from In to $M1$, and so on.

The two shaded areas in Fig.3.2 indicate the transitions which are common for the robot and the machines (loading and unloading the parts). In a complete model, these transitions should be "fused" with the corresponding transitions of machine models (Fig.3.1).

Fig.3.3 shows the refinement of net $\mathbf{M}_0$ (shown in Fig.3.2) by the two nets $\mathbf{M}_1$ and $\mathbf{M}_2$ shown in Fig.3.1; in this case $\mathcal{M} = \{\mathbf{M}_1, \mathbf{M}_2\}$, and the refinement functions are:

$$\forall\ x \in P_0 \cup T_0 : \rho(x) = \begin{cases} 1, \text{if } x \in \{t_{1a}, t_{1b}\}; \\ 2, \text{if } x \in \{t_{2a}, t_{2b}\}; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\ p \in P_0 : \phi(t_{1a}, t_{1b})(p) = \begin{cases} \{t_{1a}\}, \text{if } p = p_{10}; \\ \{t_{1b}\}, \text{if } p = p_{11}; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\ p \in P_0 : \phi(t_{2a}, t_{2b})(p) = \begin{cases} \{t_{2a}\}, \text{if } p = p_{21}; \\ \{t_{2b}\}, \text{if } p = p_{22}; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\ p \in P_0 : \psi(t_{1a}, t_{1b})(p) = \begin{cases} \{t_{1a}\}, \text{if } p = p_{11}; \\ \{t_{1b}\}, \text{if } p = p_{12}; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\ p \in P_0 : \psi(t_{2a}, t_{2b})(p) = \begin{cases} \{t_{2a}\}, \text{if } p = p_{22}; \\ \{t_{2b}\}, \text{if } p = p_{20}; \\ \text{undefined otherwise.} \end{cases}$$

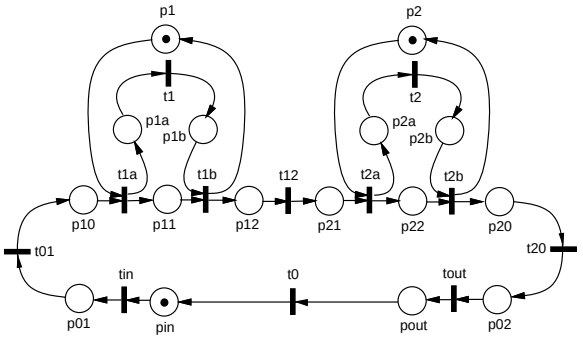

Fig.3.3. Net model of a 2–machine schedule.

In timed Petri nets 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, $\mathbf{M} = (\mathbf{N}, m_0)$, $\mathbf{N} = (P, T, A)$, and f is a firing time function which assigns the nonnegative (average) firing time $f(t)$ to each transition t of the net, $f : T \to \mathbf{R}^{\oplus}$, and $\mathbf{R}^{\oplus}$ denotes the set of nonnegative real numbers.

The behavior of a timed Petri net can be represented by a set of states and state transitions [11] which can be combined into a graph of reachable states; this graph is a semi–Markov process defined by the timed net $\mathbf{T}$. For cyclic conflict–free timed nets, such reachability graphs are simple cycles which represent the cyclic behavior of conflict–free 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 [12]. This property can be used for a very efficient structural method of performance evaluation, based on place invariants of net models [12, 14].

4. Net models of composite schedules

Petri net models of simple schedules can easily be derived from detailed sequences of the robot operations as an extension of the example given in Section 3. Fig.4.1 shows a net model of the previously (Section 2) discussed schedule A for a 3–machine cell, and Fig.4.2 shows a model of schedule D.

In composite schedules, several (possibly different) parts enter the cell in each cycle, so the models of machines must be generalized to represent possibly different operations performed on different parts by the same machine. A simple extension of the model shown in Fig.3.1 uses a free–choice structure with a branch for each part of the composite schedule. For schedules composed of pairs of simple schedules, the machine model can be as in Fig.4.3. It should be observed that this model can be created by a refinement of a single machine model (e.g., the first model shown in Fig.3.1), replacing place p_1 by another net model of a single machine (e.g., the second net model in Fig.3.1).

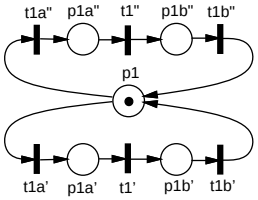

Fig.4.3. Net model of a single machine, $\mathbf{M}_1$.

A detailed sequence of robot's moves, shown in Fig.4.4, is a generalization of the model from Fig.3.2:

(t'_{in}, p'_{01}, t'_{01}, p'_{10}, t'_{1a}, p'_{11}, t'_{1b}, p'_{12}, t'_{12}, p'_{21}, t'_{2a}, p_{2c}, t_{20}, p''_{in}, t''_{in}, p''_{01}, t''_{01}, p''_{10}, t''_{1a}, p''_{1c}, t''_{21}, p''_{2d}, t''_{2b}, p''_{23}, t''_{23}, p''_{32}, t''_{3a}, p''_{31}, t''_{31}, p''_{13}, t''_{1b}, p''_{12}, t''_{12}, p''_{21}, t''_{2a}, p''_{2c}, t''_{32}, p''_{3c}, t''_{3b}, p''_{30}, t''_{30}, p''_{03}, t''_{out}, p''_{out}, t_{02}, p_{20}, t'_{2b}, p'_{23}, t'_{23}, p'_{32}, t'_{3a}, p'_{33}, t'_{3b}, p'_{30}, t'_{30}, p'_{03}, t'_{out}, p'_{out}, t'_0, p'_{in}, t'_{in})

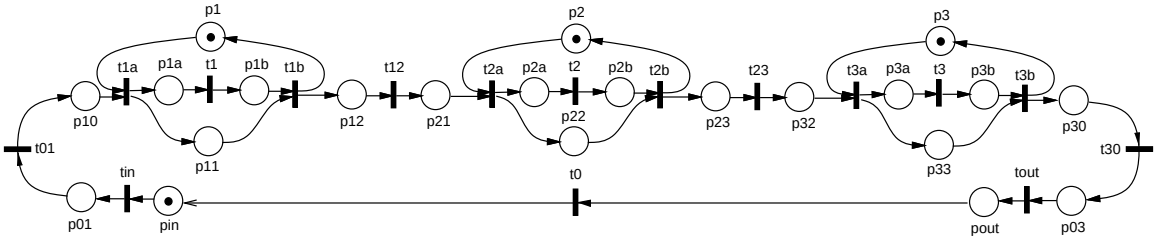

Fig.4.1. Net model of schedule A.

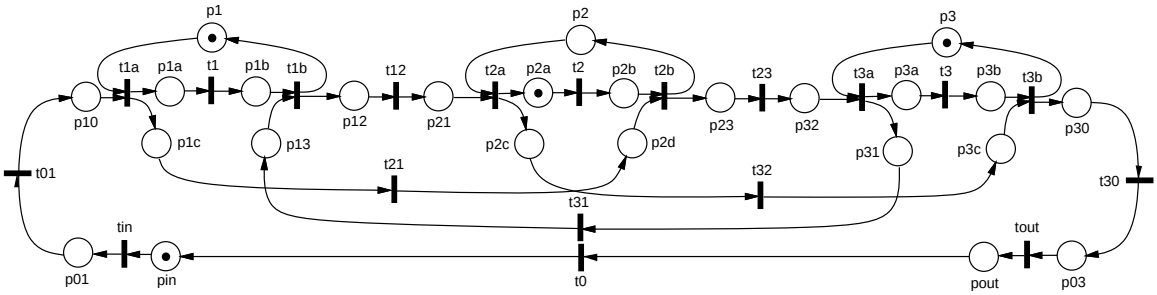

Fig.4.2. Net model of schedule D.

The shaded areas in Fig.4.4 indicate two of three subsets of transitions to be refined by the machine models $\mathbf{M}_1$, $\mathbf{M}_2$ and $\mathbf{M}_3$ (all as shown in Fig.4.3).

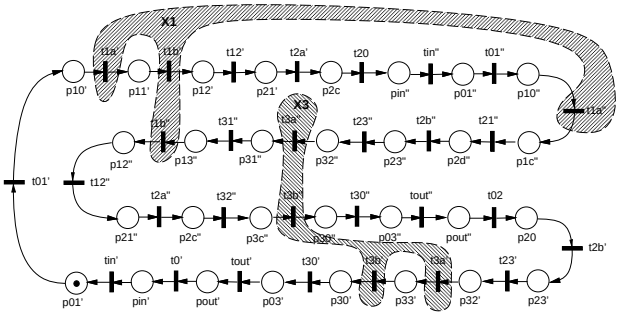

Fig.4.4. Robot's cycle for schedule A+D.

The refining functions ρ, ϕ and ψ are:

$$\forall\ x \in P_0 \cup T_0 : \rho(x) = \begin{cases} 1, \text{if } x \in \{t'_{1a}, t'_{1b}, t''_{1a}, t''_{1b}\}; \\ 2, \text{if } x \in \{t'_{2a}, t'_{2b}, t''_{2a}, t''_{2b}\}; \\ 3, \text{if } x \in \{t'_{3a}, t'_{3b}, t''_{3a}, t''_{3b}\}; \\ \text{undefined otherwise;} \end{cases}$$

Let $X_1 = \{t'_{1a}, t'_{1b}, t''_{1a}, t''_{1b}\}$, $X_2 = \{t'_{2a}, t'_{2b}, t''_{2a}, t''_{2b}\}$, and $X_3 = \{t'_{3a}, t'_{3b}, t''_{3a}, t''_{3b}\}$; then:

$$\forall\ p \in P_0 : \phi(X_1)(p) = \begin{cases} \{t'_{1a}\}, \text{if } p = p'_{10}; \\ \{t'_{1b}\}, \text{if } p = p'_{11}; \\ \{t''_{1a}\}, \text{if } p = p''_{10}; \\ \{t''_{1b}\}, \text{if } p = p''_{13}; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\ p \in P_0 : \phi(X_2)(p) = \begin{cases} \{t'_{2a}\}, \text{if } p = p''_{21}; \\ \{t'_{2b}\}, \text{if } p = p'_{2c}; \\ \{t''_{2a}\}, \text{if } p = p'_{21}; \\ \{t''_{2b}\}, \text{if } p = p''_{2d}; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\ p \in P_0 : \phi(X_3)(p) = \begin{cases} \{t'_{3a}\}, \text{if } p = p'_{32}; \\ \{t'_{3b}\}, \text{if } p = p'_{33}; \\ \{t''_{3a}\}, \text{if } p = p''_{32}; \\ \{t''_{3b}\}, \text{if } p = p''_{3c}; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\ p \in P_0 : \psi(X_1)(p) = \begin{cases} \{t'_{1a}\}, \text{if } p = p'_{11}; \\ \{t'_{1b}\}, \text{if } p = p'_{12}; \\ \{t''_{1a}\}, \text{if } p = p''_{1c}; \\ \{t''_{1b}\}, \text{if } p = p''_{12}; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\ p \in P_0 : \psi(X_2)(p) = \begin{cases} \{t'_{2a}\}, \text{if } p = p'_{22}; \\ \{t'_{2b}\}, \text{if } p = p'_{23}; \\ \{t''_{2a}\}, \text{if } p = p''_{2c}; \\ \{t''_{2b}\}, \text{if } p = p''_{23}; \\ \text{undefined otherwise;} \end{cases}$$

$$\forall\ p \in P_0 : \psi(X_3)(p) = \begin{cases} \{t'_{3a}\}, \text{if } p = p'_{33}; \\ \{t'_{3b}\}, \text{if } p = p'_{30}; \\ \{t''_{3a}\}, \text{if } p = p''_{31}; \\ \{t''_{3b}\}, \text{if } p = p''_{30}; \\ \text{undefined otherwise;} \end{cases}$$

The result of this refinement is shown in Fig.4.5 as the schedule A+D.

5. Concluding remarks

The paper shows that composite schedules for manufacturing cells can be derived by applying (generalized) net refinement operations to simple schedules. Systematic application of such refinements results in hierarchical models of schedules with a clear identification of components, so model complexity can be controlled in a better way than in 'monolithic' (or 'flat') approaches.

The models derived in the proposed way are covered by simple subnets implied by place invariants of the the

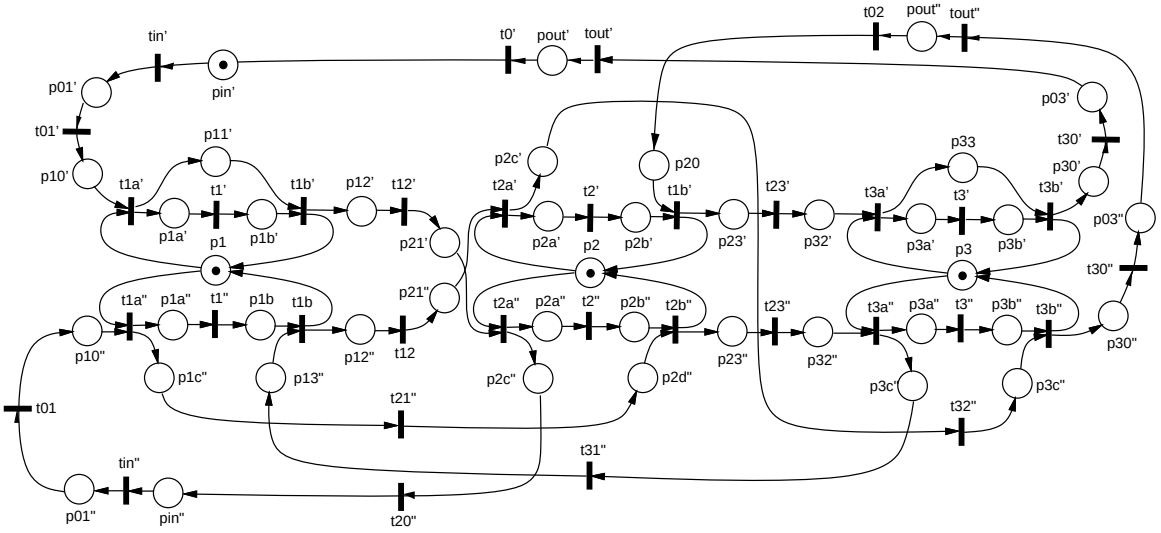

Fig.4.5. Net model of schedule A+D.

model. Performance properties of such models can be easily derived from these implied subnets [13], without the exhaustive reachability analysis.

The paper assumed that all machine operations are performed by single machines. It should be noted, however, that the same approach can be used if some operations are performed by more than one machine; the only modification is that the initial marking should assign more than one token to the corresponding machine model.

The proposed approach can be used to derive and analyze quite complex models because there is no need to deal directly with such models; since the models can be generated from a few basic components by systematic application of the refinement operations, the only information that is actually needed is how these basic components are combined together. However, more research is needed to transform these observations into practical methods.

References

[1] Ayres, R.U., Butcher, D.C., "The flexible factory revisited"; *American Scientist*, vol.81, no.5, pp.448–459, 1993.

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

[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] 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.

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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`).

[13] 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.

[14] Zuberek, W.M., "Stepwise refinements of net models and their place invariants"; Proc. 8-th Int. Workshop on Petri Nets and Performance Models (PNPM'99), Williamsburg, VA, June 1999.