1 Introduction

Theoretical models have been proposed by many authors as a useful tool for qualitative or quantitative analysis of parallel systems. Bus congestion problems, shared memory or cache conflicts, and other issues in parallel computer architectures have been analyzed using various methodologies such as queuing models[4], Markov chains[7], and extended Petri nets[8][9]. These analysis results are useful not only for evaluating existing systems but also for designing a new architecture.

However, computer designers may feel a difficulty when they use these tools especially for a large parallel machine. A model must be built not for a processing element but for the whole system. Therefore, models often become large and complicated when the number of processing units is large.

To cope with this problem, we proposed a simple yet powerful analysis method called the STMT (State Transition including Multiple Tokens) net[1]. A designer can represent a system considering only the states of an element (a processor, process, or other active elements), and some restrictions of the target parallel system. A description language (Alpha) for easy specification of the STMT net is available.

The STMT net is mathematically equivalent to a discrete Markov chain. The system described in Alpha is translated into a Markov chain for the whole parallel system. Steady state analysis is done using the transition probability matrix of the Markov chain.

State explosion is often a problem in Markov chain analysis. In the STMT net, the specified restrictions are used to minimize the number of states and consequently the analysis time. A shared memory system[12], a bus arbiter[14], and process granularity optimization [1] have been analyzed using the STMT net. Some of results are compared with results from an evaluation of a real parallel machine, and the usefulness was demonstrated.

However, through the experience of analysis with the STMT net, it appears that some systems are difficult to be described. For example, in the system including elements which behave in different manner from others, the description becomes much complicated. It comes from a feature of the STMT net that tokens in the diagram cannot be distinguished from the others.

To cope with this problem, we proposed an extension of the STMT net called the Colored-STMT (CSTMT) net in which a token can be distinguished.

First, we introduce the STMT net with a simple example. Then, the problem of the STMT net is pointed out, and the CSTMT net is presented with a practical analysis example.

2 The STMT Nets

2.1 A rough sketch of the STMT Nets

Markov chain is a powerful analysis tool for parallel machines, since it can specify precisely the state of a system including exclusive relationship between elements. In this method, state transition diagrams used for the system/hardware design are often useful to build the analysis model, and most computer designers can understand and use this tool easily. However, Markov chain requires the state transition diagram including all reachable states in the whole system and their transition probabilities. If the number of an active elements (processing units) becomes large, it is hard to built such a diagram.

If a parallel system is homogeneous, just describing a behavior of one active element is sufficient to specify the entire system. The STMT

net is a model which can specify the entire system by describing the behavior of an element and relationship between them. A state transition diagram of an active element is shared with all active elements in the system. Unlike the usual state transition diagrams, multiple tokens, which correspond to the active elements in the systems, share the same diagram of the STMT net and represent the states of the multiple elements. In order to represent interaction between tokens, *marking restrictions*, *branches* and *conditional paths* are introduced into the diagram. We will examine a simple example for introducing STMT net and CSTMT Net.

2.1.1 A broadcast memory system

We adopt a multiprocessor with a broadcast memory (BCM) as an example. The system is an old-fashioned bus connected multiprocessor that provides the BCM as a shared memory. The data written into this memory is broadcast to each processor (PU in the figure) and read out without conflict.

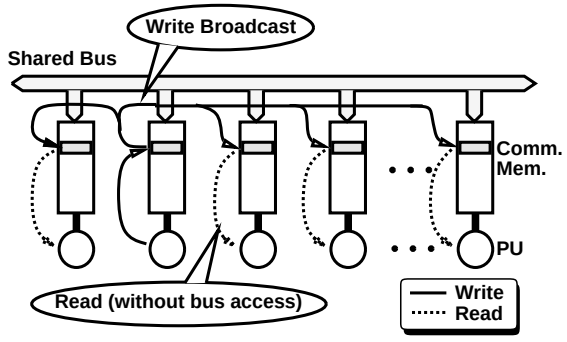

図 1: The basic model of the broadcast memory

For the sake of simplicity, both the read and write memory access require the same fixed time. Figure 2 shows the state transition diagram for each processor. Each processor changes its state by each unit time, which corresponds to the minimum interval of events

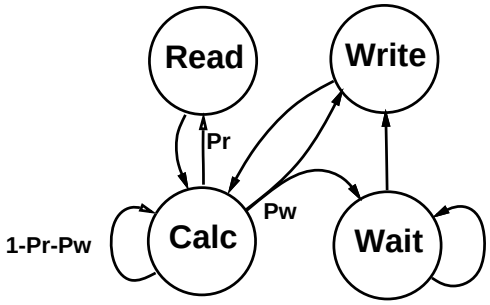

図 2: A state transition diagram for each processor in a broadcast memory system

in the system. Here, the unit time corresponds to the access time of the BCM.

A processor changes its state as follows.

- A processor that takes the state *Calc* issues a write request to the BCM with probability P_w. If there is no processor with state *Write*, it accesses the BCM and returns to the state *Calc*. Otherwise, it must change its state to the state *Wait*.
- A processor that takes the state *Calc* issues an read request to the BCM with probability P_r, and it changes its state to the state *Read*. It must keep the state *Read* until there is no processors with state *Write*.

Although a diagram for each processor like Figure 2 is easy to specify, it is difficult to write a Markov chain model for the entire multiprocessor. Since the combination of a processor state is corresponding to the state of the entire multiprocessor, the number of the state becomes great. Moreover, it is almost impossible to calculate the transient probability between states.

To cope with this problem, the STMT net is proposed. As shown in Figure 3, multiple tokens, which correspond to processors, share a state transition diagram. Interaction between

processors is represented with the *marking restriction*.

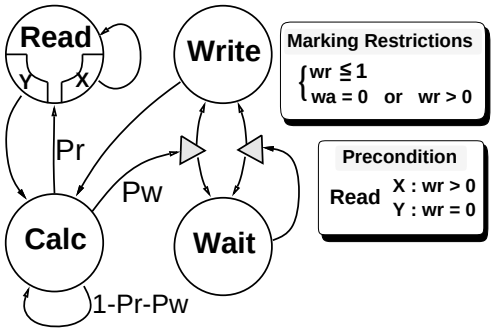

図 3: The STMT net for a broadcast memory system

We call the circle (the state in Figure 2) a *place*, and the link between places a *path*. Here, n tokens (processors) are assumed to exist in the diagram, and the number of tokens in each place is represented as follows:

wr — Tokens in *Write*.
wa — Tokens in *Wait*.
r — Tokens in *Read*.
c — Tokens in *Calc*.

Note that $wr + wa + r + c = n$. Following the terminology used in Petri-nets, we call the location of tokens in places *marking*. Each token moves from one place to another as the corresponding processor changes its state. The interaction between tokens is represented by conditions that give restrictions on the marking. These conditions are called the *marking restrictions*. Token movement that causes the violation of these restrictions is prohibited. In this example, the restrictions claim that only one token can exist in the place *Write*, and, if there are any tokens in the place *Wait*, the place *Write* cannot be empty.

Like an ordinary state transition diagram, tokens move according to the probability attached to the path. However, a path can have *branches* for multiple places only if the destination of a token can be uniquely determined by the marking restriction. In this example, a token must move to *Write* when there is no token in *Write*. Otherwise, it must move to *Wait*.

Tokens in place *Read* can move to *Calc* only if there are no tokens in place *Write*. Here, a *conditional path* is introduced to represent this situation. Probability on the conditional path can be changed by the current marking. The transition probability of the path from *Read* to *Calc* is *one* when no tokens are in *Write*, and *zero* otherwise. However, the probability of the self loop path of *Read* becomes *zero* if no tokens are in *Write*. The conditions must be exclusive of each other, and the sum of the probabilities for each condition must be *one*.

Obviously, no tokens disappear or are generated in this example. Therefore, the system state can be represented by the number of tokens in each place. In this case, the state of the system is represented with a (3-)tuple (wr, r, c), since $wa = n - wr - r - c$. Using the probabilities and the given restrictions, the transition probability from a state (wr_i, r_j, c_k) to (wr_l, r_m, c_n) can be calculated. This means that the STMT net can be transformed into a discrete Markov chain and, thus, it is easy to analyze the steady state of the system as well.

Results from this model are compared with the data from a real multiprocessor called ATTEMPT-0 [3] which has the BCM as its shared memory. The transition probabilities in the STMT net are set based on the execution of a single processor. The target applications are the following parallel CAD programs:

- *ESIM* [2] : An electric circuit simulator.
- *LOGIQUE* [6] : A parallel logic simulator.

Figure 4 and Figure 5 show the bus utilization ratios acquired from the system analysis using the STMT net and the evaluation on

ATTEMPT-0. Using the STMT net, we obtained analytical results that are close to those from real executions.

In Figure 4, the bus utilization from the STMT net is slightly higher than that in the real execution. This is because the algorithmic idle time of the processors caused by load imbalance and synchronization is not modeled.

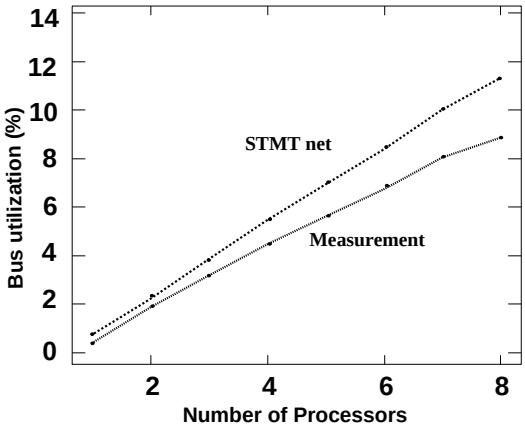

図 4: The bus utilization ratio (ESIM)

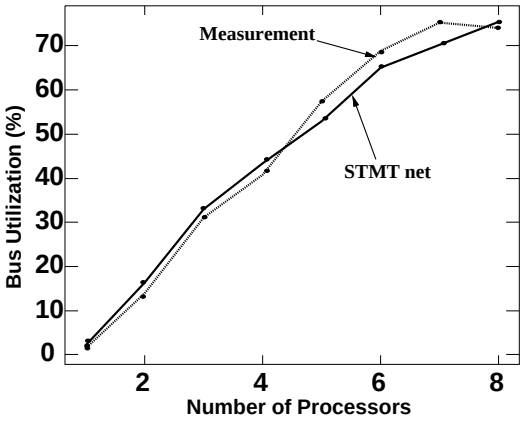

図 5: The bus utilization ratio (LOGIQUE)

3 The Colored STMT Net

3.1 The problem of the STMT nets

As demonstrated with an example in the previous section, homogeneous systems are easily described with the STMT net. However, if the system includes an element which behaves in the different manner, the description becomes much complicated. It comes from the feature of the STMT net that tokens in the diagram cannot be distinguished from the others. For example, in a multiprocessor system, the job of processors are often different each other. In such a case, each token corresponding to a processor moves with the different probability depending on its allocated job.

Moreover, in parallel systems, the identification of tokens is often required even if the target system is seamed to be homogeneous.

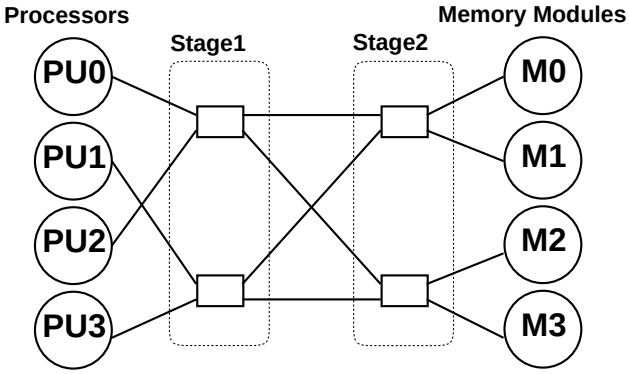

図 6: A diagram of the MIN connected multiprocessor.

Figure 6 shows a simple multiprocessor in which each processor is connected to shared memory modules with a Multistage Interconnection Network (MIN). For the sake of simplicity, there is no buffers in each switching element in the MIN. When a conflict occurs, one of the conflicting packets is correctly routed, and the other is temporarily discarded for being inserted again. A pro-

cessor issues an access request to each memory module M_0,M_1,M_2,M_3 with probability P_0,P_1,P_2,P_3 respectively.

The following nine states are required for each processor:

- $Calc$: the processor is under calculation, and does not access shared memory modules.
- $M_0, \cdots, M_3$: the processor is reading data from shared memory modules $M_0, \cdots, M_3$.
- $W_0, \cdots, W_3$: the processor is waiting for the memory module being available.

Figure 7 shows the STMT net for this multiprocessor system.

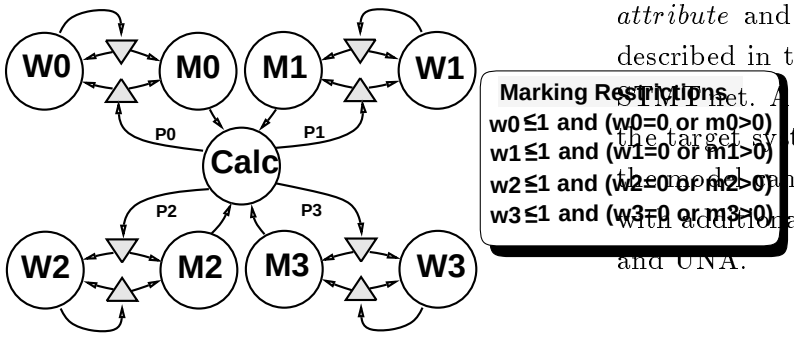

図 7: The STMT net for the MIN connected multiprocessor.

However, this model cannot include the description about conflict between packets occurred in the MIN. Each token corresponding to a processor must be distinguished because the location of processor and memory module is required to decide whether packet is discarded.

There are many other parallel systems in which each element behaves basically in the same manner but must be distinguished. To cope with this problem, we proposed an extension of the STMT nets called the Colored-STMT (CSTMT) nets in which an attribute is associated to a token.

3.2 A rough sketch of the Colored STMT Net

In order to describe the behavior of a token precisely, each token is associated with an *attribute* in the CSTMT net. To distinguish a token with the same *attribute*, a token is allowed to have an integer number *Unique Number of the Attribute* (UNA). The *attribute* shows the property of tokens, and the behavior of tokens with the same *attribute* is described in a lump. The UNA is only used when tokens with the same *attribute* are required to be distinguished. The UNA is sometimes used to decide the priority of the token.

Except for the description about the *attribute* and UNA, the CSTMT net can be described in the same manner as the original STMT net. A user can specify a rough model of the target system in the STMT net first, then, the model can be modified to the CSTMT net with additional description about the *attribute* and UNA.

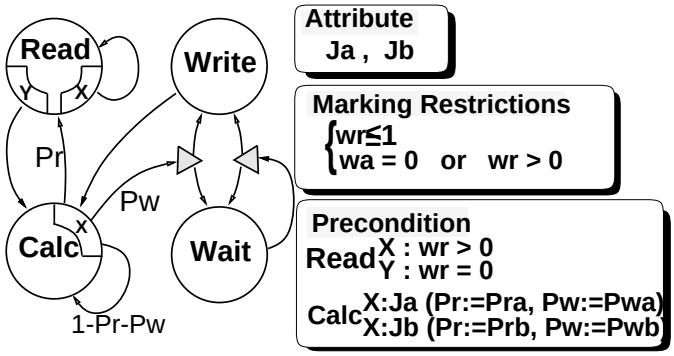

図 8: The colored STMT net for the broadcast memory system with heterogeneous load.

Figure 8 shows the CSTMT net for the multiprocessor with the BCM which is analyzed in Section 2. Unlike the model shown in Figure 2, two different types of job (job A and job B) are mapped into processors. Here, *attribute*

J_a and J_b show which job is allocated to the processor (token).

Tokens whose attribute is J_a move with the probability P_{ra} and P_{wa}, when they are in the place $Calc$. Similarly, tokens whose attribute is J_b move with the probability P_{rb} and P_{wb}. This specification is described in the *precondition* by using the *conditional path*.

Here, we show more complicated example, the MIN connected multiprocessor (Figure 6) which cannot be described by the original STMT nets. For the precise modeling, the topology of the network must be fixed. Here, the MIN is assumed to be an omega network as shown in Figure 6.

In Figure 6, packets conflict at each stage when the following conditions are satisfied:

$$\begin{array}{ll} Stage1: & p_a \bmod 2 = p_b \bmod 2 \text{ AND } m_a/2 = m_b/2 \\ Stage2: & m_a = m_b \end{array} \quad (1)$$

where symbol '/' is an integer division.

In this case, the UNA is used for identification of the Processing Unit (PU) number. The condition of the packets conflict can be described as the additional marking restriction using the UNA.

$$\begin{array}{l} for(i = 1; i \leq 4)\{ \\ \quad (M_0.PU(i \bmod 2) + M_1.PU(i \bmod 2) \leq 1) \text{ and} \\ \quad (M_2.PU(i \bmod 2) + M_3.PU(i \bmod 2) \leq 1) \\ \} \end{array} \quad (2)$$

where let $M_i.PU(j)$ be the number of tokens whose UNA is j in the place M_i.

3.3 Mathematical Specification

In this section, we present a formal specification of the STMT net and CSTMT Net. For the sake of simplicity, we assume that the net do not have branches. The complete specification of the STMT net is given in [1].

3.3.1 Definition of the STMT Nets

The STMT net is a five-tuple

$$\text{STMT net} = \{\mathbf{P_L}, \mathbf{P_A}, \mathbf{R}, M_0, \mathbf{P_R}\} \quad (3)$$

where

- $\mathbf{P_L}$ The set of places :

$$\mathbf{P_L} = \{pl_1, pl_2, pl_3, \cdots, pl_n\} \quad (4)$$

- $\mathbf{P_A}$ The set of paths :

$$\mathbf{P_A} \subset \mathbf{P_L} \times \mathbf{P_L} \quad (5)$$

An element of $\mathbf{P_A}$ is represented by pa_{ij}, which is a path from a source place pl_i to a destination place pl_j. The index set of pa_{ij} is denoted as $\mathbf{I_{PA}}$.

- $\mathbf{R}$ Marking restrictions:
Marking restrictions are the set of markings that are allowed in the system. Here, a marking M_q is a set of the number of tokens in each place and is denoted as

$$M_q = \{m_{q1}, m_{q2}, \cdots, m_{qn}\}, \quad (6)$$

where m_{qi} is the number of tokens in the place pl_i.

- M_0 The initial marking:

$$M_0 = \{m_{01}, m_{02}, \cdots, m_{0n}\} \in \mathbf{R} \quad (7)$$

Here, the sum of the tokens of the initial marking is represented by N (N is constant):

$$N = \sum_{j=1}^{n} m_{0j} \quad (8)$$

This expression represents that tokens are never generated or deleted.

- $\mathbf{P_R}$ The set of transition probabilities associated with paths:

$$\mathbf{P_R} = \{pr_{ij} | (i, j) \in \mathbf{I_{PA}}\} \quad (9)$$

Tokens move between places according to these probabilities. The transition probability is a function of the current marking M_q.

$$0 \leq pr_{ij} \leq 1, \qquad (i,j) \in \mathbf{I_{PA}} \tag{10}$$

$$\sum_{\{j|(i,j)\in \mathbf{I_{PA}}\}} pr_{ij} = 1 \tag{11}$$

Here, the movement of tokens in the STMT net is represented by a change of the marking according to $\mathbf{P_R}$.

In the STMT net, a system state is represented by a marking, and the current marking is determined probabilisticly from the previous marking. Therefore, if the transition probability from a marking to another is obtained, it can be reduced into a discrete-time Markov chain. The detail of the conversion algorithm is shown in [1].

3.4 Definition of the Colored STMT Net

The colored STMT net is a six-tuple

$$\text{STMT-net} = \{\mathbf{P_L}, \mathbf{P_A}, \mathbf{R}, M_0, \mathbf{P_R}, \mathbf{L_A}\}$$

where

$\mathbf{P_L}$	A set of places	: $\mathbf{P_L} = \{pl_1, pl_2, \cdots, pl_n\}$
$\mathbf{P_A}$	A set of directed paths	: $\mathbf{P_A} = \{pa_{11}, pa_{12}, \cdots, pa_{nn}\}$
$\mathbf{L_A}$	A set of labels	: $\mathbf{L_A} = \{la_1, la_2, \cdots, la_m\}$.

The only difference from STMT net is that each token has one of labels in $\mathbf{L_A}$. A label is decided from the *attribute* and UNA. Tokens belong to the same attribute can be distinguished with the UNA, but tokens with the same label cannot be distinguished.

A set of attributes is:

$$A_t = \{at_1, at_2, \cdots, at_k\}. \tag{12}$$

A set of labels $\mathbf{L_A}$ is represented by:

$$\mathbf{L_A} = \{la | la \in \{i\} \times UNA_i,\ i \in A_t\} \tag{13}$$

where

$$UNA_i = \{un | un \in \text{Natural Number}, un \leq num(i)\}. \tag{14}$$

Here, $num(i)$ is the number of tokens with the *attribute* i when the UNA is used. If the UNA is not used, $num(i)$ becomes *one*.

The sum of tokens is represented with N:

$$N = \sum_{i=1}^{m} n_i \tag{15}$$

where n_i is the number of tokens whose label is la_i.

Now, assuming that there are only tokens with label la_i in the system, a marking is represented by:

$$M_q^{(i)} = \{m_{q1}^{(i)}, m_{q2}^{(i)}, \cdots, m_{qn}^{(i)}\}. \tag{16}$$

$m_{qj}^{(i)}$ is the number of tokens with label la_i in the place pl_j. Thus, a marking of the system, in which there are tokens with label $la_1, la_2, \cdots, la_m$, is represented by:

$$M_q = \{M_q^{(1)}, M_q^{(2)}, \cdots, M_q^{(m)}\}. \tag{17}$$

The marking restriction $\mathbf{R}$ is a set of markings which is allowed in the system. An initial marking M_0 is:

$$\begin{aligned} M_0 &= \{M_0^{(1)}, M_0^{(2)}, \cdots, M_0^{(m)}\} \in \mathbf{R} \\ M_0^{(i)} &= \{m_{01}^{(i)}, m_{02}^{(i)}, \cdots, m_{0n}^{(i)}\} \end{aligned} \tag{18}$$

where

$$\begin{aligned} n_i &= \textstyle\sum_{j=1}^{n} m_{0j}^{(i)} \\ N &= \textstyle\sum_{i=1}^{m} \sum_{j=1}^{n} m_{0j}^{(i)}. \end{aligned} \tag{19}$$

$\mathbf{P_R}$ is a set of transition probabilities associated with paths:

$$\mathbf{P_R} = \{pr_{11}, pr_{12}, \cdots, pr_{nn}\}. \tag{20}$$

The transition probability is a function of both the current marking M_q and the label of the token. A token moves between places according to this probability decided from the marking and its own label.

3.4.1 Conversion to the STMT net

The CSTMT net which includes only a label ($m = 1$) is equivalent to the original STMT net. Therefore, the CSTMT net can be converted to the STMT net by copying a set of places $\mathbf{P_L}$ so as to exist only tokens with the same label in the $\mathbf{P_L}$.

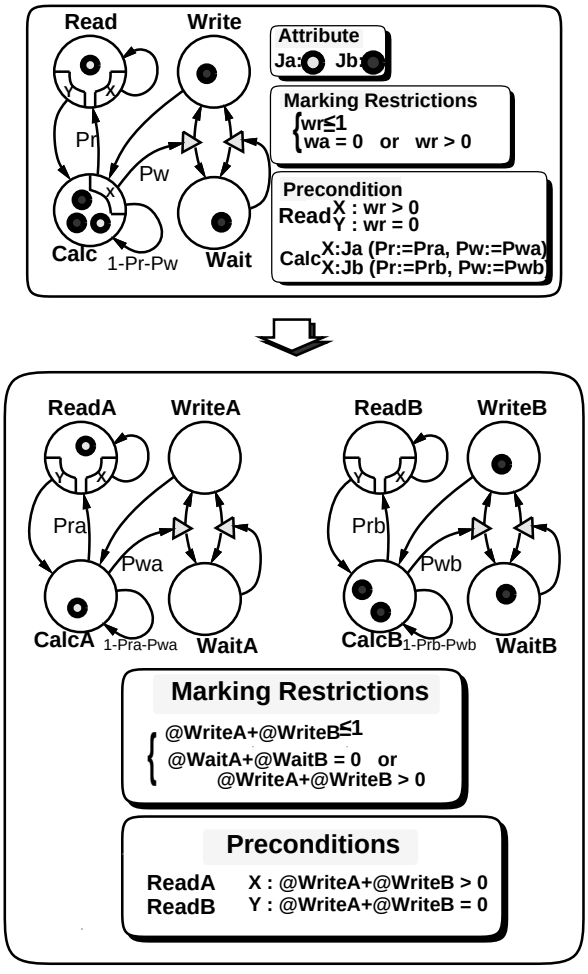

図 9: The reduction of the colored STMT net to the STMT net

Figure9 shows an example of the reduction. Like Figure8, a broadcast memory system of a multiprocessor with different jobs (Ja and Jb) is represented by the CSTMT Net. In this case, the structure of the network is duplicated into dedicated networks for processors with the job A and ones with the job B. Marking restrictions and preconditions are also rewritten for these two networks.

In general, by copying a set of places $\mathbf{P_L}$ for every label, the CSTMT net$\{\mathbf{P_L}, \mathbf{P_A}, \mathbf{R}, M_0, \mathbf{P_R}, \mathbf{L_A}\}$ can be converted to another CSTMT net$\{\mathbf{P_L}', \mathbf{P_A}', \mathbf{R}', M_0', \mathbf{P_R}', \{la\}\}$ which is equivalent to the STMT net.

Here, consider a set of place $\mathbf{P_L}_i$ which includes only tokens with the same label la_i.

$$\mathbf{P_L}_i = \{pl_{1i}, pl_{2i}, \cdots, pl_{ni}\} \tag{21}$$

A set of places for the whole system is:

$$\mathbf{P_L}' = \mathbf{P_L}_1 \cup \mathbf{P_L}_2 \cup \cdots \cup \mathbf{P_L}_m = \{pl_{11}, pl_{21}, \cdots, pl_{nm}\}. \tag{22}$$

Thus, the number of places is $n \times m$. A marking M_q' is represented only by a set of token number in each place because tokens in the $\mathbf{P_L}_i$ must have the label la_i.

$$M_q' = \{m_{q11}, m_{q21}, \cdots, m_{qnm}\} \tag{23}$$

Since the total number of tokens in $\mathbf{P_L}_i$ must be n_i, a set of available marking $\mathbf{R}^*$ is:

$$\mathbf{R}^* = \{M_q' | \sum_{j=1}^{n} m_{qji} = n_i\}. \tag{24}$$

Here, $\mathbf{R}'$(a set of markings allowed in the system) is a subset of $\mathbf{R}^*$.

Tokens can't move from a place $\mathbf{P_L}_i$ to a place $\mathbf{P_L}_j$ $(j \neq i)$ because the label of a token is not changed. Thus, $\mathbf{P_R}'$(a set of transition probabilities) is:

$$\begin{aligned} &\mathbf{P_R}' = \{pr_{(11)(11)}, pr_{(11)(21)}, \cdots, pr_{(nm)(nm)}\} \\ &pr_{(ix)(jy)} = 0 \qquad\qquad (i \neq j) \end{aligned} \tag{25}$$

where let $pr_{(ix)(jy)}$ be the probability of a transition from a place pl_{ix} to a place pl_{jy}.

These definitions show that a CSTMT net can be analyzed by converting to the corresponding STMT net.

4 A practical example

Using the CSTMT net, target systems can be precisely modeled and evaluated.

As a practical example, the arbiter of the IEEE
Futurebus is analyzed here. Futurebus[10] is designed as a standard bus used in multiprocessors. The arbitration protocol of Futurebus specifies a distributed scheme for selecting only one module (usually a processor) as a bus master.

According to a fixed unique priority number of each module, the module with the highest priority number in the modules requesting for a mastership can be a bus master. Using the fixed priority arbiter, a weak module may not get the mastership continuously prevented by stronger modules. This situation is called the starvation. In Futurebus, to avoid the starvation, a module which takes a mastership becomes a "fairness" module, and is prohibited to get a mastership until no other modules request the mastership. This mechanism is called the fairness release.

Here, we apply the CSTMT net and analyze the influence of the fixed priority strategy to the performance.

A distributed hardware arbiter is provided for competing, and each module takes one of the following states:

- current master(CM): The module holds the bus mastership.
- free bystander(FB): The module does not require the bus mastership and can be participant in the competing.
- inhibited bystander(IB): The module does not require the bus mastership, and the participation in the competing is prohibited to avoid the starvation of other modules.
- competitor(C): The module requires the bus mastership and is a participant in the competing.
- withholder(W): The module requires the bus mastership but the participation in the competing is prohibited to avoid the starvation of other modules.

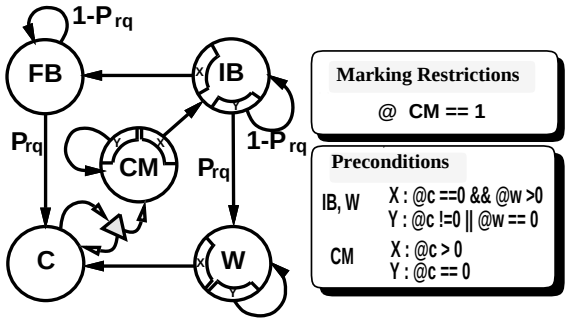

図 10: The STMT net for the arbitration protocol of Futurebus.

As shown in Figure 10, these five states correspond to places in the STMT net, and the modules are represented by tokens. The time of the control acquisition cycle is used as the unit time of the system. Here, the number of tokens in each place $\{FB, C, CM, IB, W\}$ is represented by small letters $\{fb, c, cm, ib, w\}$, and the total number of tokens is N ($fb + c + cm + ib + w = N$). The number of bus masters must be one (i.e., $cm = 1$). Here, we assume that a module issues a request for mastership with probability P_{rq} for each unit time. In the CSTMT Net, each token in the place FB and IB moves to C and W, respectively, with probability P_{rq}.

When there are no competitors, all tokens in IB and W are permitted to participate in the competing. In the CSTMT Net, tokens in W and IB move into C and FB, respectively, if $c = 0$. Here, the UNA is used as the fixed priority of the arbitration. A token with the

highest priority in place C is selected and sent to place CM.

For easy specification of the CSTMT net, a description language called Beta is prepared.

Figure11 is a description corresponding to the bus arbitration protocol shown in Figure10. The target system is described using two parts: the state description part and the transition description part. In the former (line 1-8 of Figure11), the static features of the system, such as the number of tokens, names of places, and marking restrictions are described. In this example, the *attribute* Mdl is attached to each token with the UNA ($UNA < Max$) in the line 1.

In the latter (line 10-30 of Figure11), the paths between places are defined with their associated weights which correspond to the transition probabilities. In Beta, a path can be selected by the UNA of tokens as shown in the line 14-17.

Beta is upper compatible with the description language Alpha for the STMT net [1]. A model described in Beta is converted into Alpha description with a preprocessor. Using the STMT analyzer, the is further converted into the corresponding Markov chain, and the expected number of tokens in each place at the steady-state is automatically calculated.

Here, the system with 8 modules is analyzed as an example. Figure 4 shows the probability of the master request P_{rq} versus the working ratio of a processor. Here, we call the module with the highest priority as the strongest module and one with the lowest priority as the weakest module. With a large P_{rq}, the frequency of a module issuing the arbitration request increases, and the loss of the arbitration becomes large. The difference between the strongest module and the weakest module also becomes large. This figure shows that the fixed priority of Futurebus causes a certain degree of unfairness between modules.

5 Conclusion

By associating *attribute* and UNA to tokens in the STMT net, the power of modeling is much enhanced in the CSTMT net. Systems which are difficult to be represented by the STMT net including a multiprocessor with unbalanced job, a MIN connected multiprocessor and a priority arbiter can be precisely modeled and analyzed with the CSTMT net. The model described in the language Beta is converted to a discrete-time Markov chain, and the steady-state of the system is automatically analyzed.

Although the state number of Markov chain becomes large, the computation time of the CSTMT net can be estimated in advance. Users can make the model considering the computation time for the analysis.

参考文献

[1] T.Kimura, O.Yamamoto, H.Amano, T.Takemoto, "The STMT net: An analysis Model for parallel systems," Proc. of Summer Computer Simulation Conference, pp.319-327, Jul. 1992.

[2] H. Amano, "RSM (Receiver Selectable Multicast): A Communication mechanism for multiprocessors," Proc. of 2nd International Conference on Computers and Applications, pp.149–156, June 1987

[3] H. Amano, T. Terasawa, and T. Kudoh, "Cache with Synchronization Mechanism," Proc. of 11th IFIP World Computer Congress, pp.1001–1006, August 1989

[4] P.J.Denning and J.P.Buzen, "The operational analysis of queuing network models," ACM Computing Surveys, Vol.10, pp.225–261, September 1978.

[5] Mark A. Holliday and Mark K. Vernon, "A Generalized Timed Petri Net Model for Performance Analysis", IEEE Transactions on Software Engineering, Vol.SE-13, No.12, pp.1297–1310, December 1987.

[6] T.Kudoh, T.Kimura, H.Amano, and T.Terasawa, "A parallel logic simulation algorithm based on query," International Conference on Parallel Processing, Vol.III, pp.262–266, August 1992.

[7] M.Ajmone Marsan, G.Balbo, and G.Conte, "Comparative performance analysis of single-bus multiprocessor architecture," IEEE Transactions on Computers, Vol.C-31, pp.1179–1191, December 1982.

[8] M.Ajmone Marsan, "Stochastic Petri nets: An elementary introduction," Advances in Petri Nets 1989, Lecture Notes in Computer Science, Springer Verlags, pp.1–29, 1989.

[9] M.K.Vernon, and M.A.Holliday, "Performance Analysis of Multiprocessor Cache Consistency Protocols Using Generalized Timed Petri Nets," Proc. of Performance 86 and ACM Sigmetrics 1986 Joint Conf. on Computer Performance Modeling, Measurement, and Evaluation, Taleigh, N.C., pp.9–17, May 1986.

[10] "IEEE Standard Backplane Bus Specification for Multiprocessor Architectures: Futurebus," IEEE June 1988

[11] M.Ajmone Marsan, G.Balbo, and G.Conte: "Comparative performance analysis of single-bus multiprocessor architecture", IEEE Trans. Comput., **C-31**, pp.1179-1191 (Dec.1982).

[12] S.Torii, T.Takemoto, H.Amano, S.Ogura: "A performance evaluation of communication memory system for parallel machines with a shared bus", Transactions of Information Processing Society of Japan **33**, 3, pp.307-319 (1992-3) *in Japanese.*

[13] T.Kimura, O.Yamamoto, H.Amano, T.Takemoto: "STMT-Net: An Analysis Model for Parallel Systems" Trans. IEICE (Institute of Electronics,Information and Communication Engineers) ,D-I,Vol.J75-D-I,No8, pp.654-663 (1992-8) *in Japanese.*

[14] O.Yamamoto, S.Torii, H.Amano: "A performance evaluation of the arbiter of IEEE standard backplane bus: Futurebus", Information Processing Society of Japan 91-ARC-14,pp.111-118 (1991-11) *in Japanese.*

```
        /* Definition of attributes */
1   %token          Mdl(Max)  /*  attribute: Mdl, UNA<Max  */

        /* Definition of Places */
2   %place          FB            /*  Free bystander */
3   %place          C         /*  Competitor    */
4   %place          CM        /*  Current Master */
5   %place          IB        /*  Inhibited bystander  */
6   %place          W         /*  Withholder  */

    /* Global restrictions */
7   %restrict       (@CM == 1) /* There must be a current master *
8   %%
9
        /* Normal paths and their probabilities */
10  FB      :       C       Prq                         /* Request
11          |       FB      1-Prq
12          ;
13
        /* Conditional paths whose probabilities depend on the att
14  C  Mdl(Max)                                 :  C , CM /* Highest
15     Mdl(i:=1..Max-1)<<@C.Mdl(i..Max)==0>> :  C , CM /* Priority
16                      <<else>>                :  C
17                                              ;
18
        /* Conditional paths whose probabilities are independent w
19  IB <<@C ==0 && @W > 0>> :         FB              /* Fairness
20     <<else>>             :         W        Prq
21                          |         IB       1-Prq
22                          ;
23
24  W <<@C ==0 && @W > 0>>  :         C               /* Fairness
25    <<else>>              :         W
26                          ;
27
28  CM <<@C > 0 >>          :         IB              /* Waiting
29     << else >>           :         CM
30                          ;
```

図 11: The description of the bus arbitration protocol of Futurebus in Beta

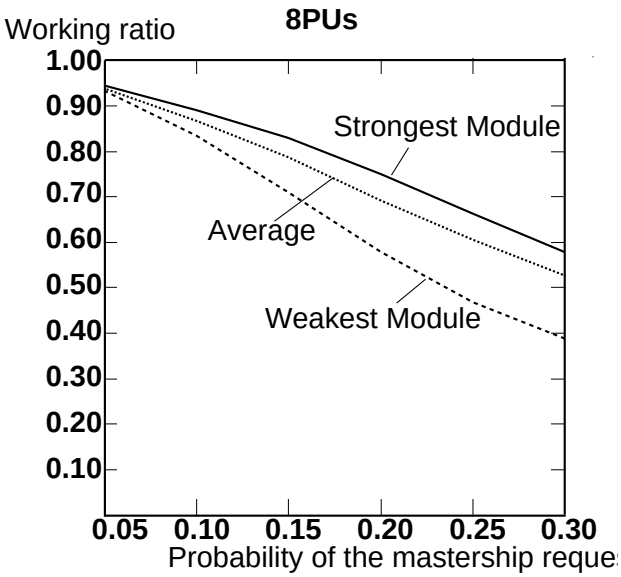

図 12: The module working ratio