計算機アーキテクチャ 96－22
(1992. 10. 23)

並列システム性能解析モデルであるSTMT-netのcolored化

竹本 卓, 木村 哲郎, 山本 欧, 天野 英晴

慶應義塾大学 理工学部

あらまし 単純であるが幅広い用途を持つ理論解析モデルであるSTMTネットを拡張し、トークンに属性を持たせたColored STMTネットを提案する。Colored STMTネットの導入により、STMTネットでは記述が困難であった幾つかのシステムに表現を与えることが可能となった。専用記述言語Betaで記述されたColored STMTネットはプリプロセッサにより等価なSTMTネットに変換され、更にSTMTアナライザにより対象とするシステム全体のマルコフチェーンモデルに変換される。そして通常のマルコフ解析同様の手法で、システムの安定状態での挙動を知ることができる。Colored STMTネットを用いて、IEEE Futurebusのアービトレーションプロトコルについて解析を行なった。その結果、バスマスタ要求率の変化とアービトレーションの公平性との間の関係が明らかになった。

和文キーワード 遷移確率行列, マルコフ解析, 並列システム

The Colored STMT Net:An Analysis Model for Parallel Systems

T.Takemoto , T.Kimura , O.Yamamoto , H.Amano

Faculty of Science and Technology, Keio University
Yokohama, 223, Japan

Abstract The STMT net is a simple yet powerful analysis model. However, parallel systems including elements with attribute cannot be treated by the model. To cope with this problem, the Colored STMT net, which provides attributes for each tokens in the STMT net, is proposed. The system described in *Beta* (The description language for Colored STMT net) is translated into an equivalent STMT net by the preprocessor, and the Markov chain of the whole parallel system is generated by the STMT net analyzer. Then steady state of the system is analyzed. In order to demonstrated the power of the Colored STMT net, the analysis result of arbitration protocol of the IEEE Futurebus. The result demonstrates that the fairness of getting the bus mastership is severly degraded with the fixed priority strategy of the bus protocol.

英文 key words transition probability matrix, markov chain, parallel system

1 はじめに

新しい並列システムのアーキテクチャ設計のためには、そのアーキテクチャがさまざまな条件のもとでどのような挙動を示すかを正しく解析, 評価することが重要である。STMT(State Transition including Multiple Tokens) ネット [4] は単純であるが幅広い用途を持つ並列システム解析モデルである。STMT ネットでは、並列システム内の個々の要素 (プロセッサ、プロセス等) をトークンとみなし、その状態遷移図とトークン間の相互作用を表す制約条件とを用いてシステム全体の動作を記述する。STMT ネットは、すでに共有メモリシステム [3][4], バスアービトレーションプロトコル [5], プロセス粒度の最適化 [4] 等の問題に適用されその有効性が確認されている。

しかし STMT ネットでは、全てのトークンを同一のものとして扱っているため、トークンに優先順位を設定するといった場合の記述は非常に困難である。この問題を解決するため、本論文では各トークンに属性を与えることを可能にした Colored STMT ネットを提案し、従来の STMT ネットでは記述が困難であった幾つかのシステムに対し Colored STMT ネットに対して適用し評価を行なった。また、トークンに属性を与えたことによるシステムの状態数の増加と計算コストの増加について考察を行なった。

2 理論解析モデル STMT ネット

2.1 マルコフ解析の問題点

マルコフ解析は、システムのとり得る状態と遷移確率を図示する状態遷移図とを用いてシステムの挙動を記述、解析する方法であり、解析対象となるシステムの要素間の同期や協調動作を正確に記述できる点で、並列システムの解析に適している [1][2]。しかし、一般に並列システムでは並列に動作する要素の数が多いため、状態数が膨大になり、記述に多大な労力を要する。

例として、ブロードキャストメモリを共有メモリとするバス結合型マルチプロセッサシステムに対するマルコフ解析を考える。ブロードキャストメモリでは、同時に複数のプロセッサによる読出しが可能であるが、書き込みは同時に一つのプロセッサのみ可能である。また、読出し要求は書き込みアクセスによりブロックされる。各プロセッサはローカルなデータを用いて計算を行なっている状態 (Calc)、共有メモリに対して読み出しを行なっているか、または他のプロセッサの書き込みによって読み込み要求がブロックされている状態 (Read)、共有メモリに対して書き込みを行なっている状態 (Write)、他のプロセッサの書き込みによって、書き込み要求がブロックされている状態 (Wait) の 4 状態を持つ。図 1にシステム内の 1 つのプロセッサについての状態遷移図を示す。Calc 状態にいるプロセッサは共有メモリに対して確率 P_w で書き込み要求、確率 P_r で読出し要求を出す。複数の書き込み要求が同時に発生すると、1 プロセッサのみが Write 状態になり、他のプロセッサは Wait 状態になる。

システム全体の状態は、それぞれの状態をとるプロセッサ数の分布によって表すことができる。Calc,Read,Write,Wait の各状態をとるプロセッサの数をそれぞれ c, r, wr, wa、全プロセッサ数を n とすると、

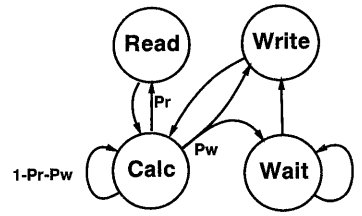

図 1: ブロードキャストメモリにおける各プロセッサの状態遷移図

$c+r+wr+w=n$ を満たす正の整数の組 (c, r, wr, wa) の組合わせの数だけマルチプロセッサの状態が存在することになる。この数は $(n+3)!/(3!n!)$ となり, n^3のオーダで増加する。このことは、プロセッサ数が 4〜8 台程度の比較的小規模なシステムですら、そのとり得る状態数は非常に多くなることを意味する。このような状況で、あるプロセッサの書き込みが他のプロセッサをブロックするという相互作用を考慮に入れ、システムの各状態間の遷移確率を求めることは困難である。

2.2 STMT ネット

上で述べたように、相互作用を持つ複数の要素からなるシステム全体の状態遷移は非常に複雑なものとなる。しかし、図 1で示したような、個々の要素についての状態遷移図を描くことは容易である。そこで、個々の要素の振舞いを表現する一つの状態遷移図と、要素間の相互作用を表す制約条件等を用いれば、システム全体の振舞いを容易に表現することが可能になる。要素の状態遷移を状態遷移図中のトークンの移動で表すと、システムの動作は一つの状態遷移図中を各要素に相当する複数のトークンが相互作用をもちつつ移動するものとして表現できる。以上によりシステムを記述する表記法が STMT ネットである。

STMT ネットはマルコフチェーンに等価であり、STMT ネットによる表現からシステム全体の状態遷移確率行列を導出することができる。したがってマルコフ解析と同様の手法で、システムの解析が行なえる。

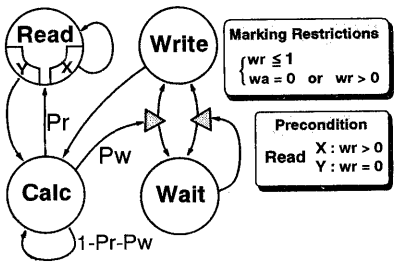

図 2: ブロードキャストメモリに対する STMT ネット

図 2に、先のブロードキャストメモリの STMT ネットを示す。図中の状態を表す円をプレース、プレース間を結ぶ有向辺をパスと呼ぶ。パスに付属している文字は遷移確率を表す。プロセッサに対応するトークンは、システム内で一定の時間刻みに同期してプレース間をパスを通って移動する。

トークンの分布状態をマーキングと呼ぶ。トークン間の相互作用は、マーキングに関する制約条件(マーキング制約)で与えられ、この条件に反するトークンの移動は禁止される。この例でのマーキング制約は次のようになる。

・プレース Write には二つ以上のトークンは存在できない。

・プレース Wait にトークンが存在する場合は、プレース Write にトークンが必ず存在する。

図中でマーキング制約は Markig Restriction の欄に記述されている。

トークンは各パスに与えられた遷移確率に従って移動するが、移動先のプレースがマーキング制約によって唯一に決まる場合、遷移確率は意味を持たなくなる。STMT ネットでは、このようなトークンの移動を分岐付きパス(△記号をもつパス)で記述する。例では Calc から Write または Wait への遷移に分岐付きパスが用いられている。この分岐付きパスは、Wait にトークンがいない場合は Calc にあるトークンが確率 P_w に従って Write に移動し、Write にトークンがある場合は確率 P_w に従って Wait に移動することを表している。

一方、トークン間の相互作用はマーキング制約だけでは記述できない場合がある。図 2において Read 状態にあるプロセッサが読出しを完了して Calc 状態に戻るのは、Write 状態にあるプロセッサが存在しない場合に限られる。このような制約は条件付きパスを使って表現する。条件付きパスは、現在のマーキングがある条件を満たす時のみ、その遷移が有効となるパスである。図 2では Read 内から出てゆくトークンは、Write にトークンがある場合は Read に留まり、Write にトークンがいない場合は Calc に移動する。プレース内の扇型は、そこを始点とするパスが有効となる条件に対応しており、扇型に与えられたラベルは Precondition 欄中の条件記述との対応に用いられる。

STMT ネットの実際の記述は、専用記述言語 Alpha [4] によって行なう。Alpha で記述された STMT ネットはアナライザにより、対象とするシステム全体のマルコフチェーンモデルに変換され、遷移確率行列が計算される。

3 Colored STMT ネット

3.1 STMT ネットの問題点

STMT ネットは、並列システム内の個々の要素の動作の規則が、多くの場合同じであるとみなせることから、一つの状態遷移図をすべての要素が共有する表示法を採用している。言い換えれば各トークンは同一であり区別することができない。したがって、各要素の動作が一部分でも異なる部分があると、システムの記述はたちまち困難になる。

例として、図 3に示すようなプロセッサと共有メモリの間がブロッキング型の多段結合網 (Multistage Interconnection Network:MIN) で結合されたマルチプロセッサシステムを考える。説明を簡単にするため、結合網内のエレメントスイッチにパケットのバッファがない MIN を仮定する。すなわち、結合網内で衝突が生じた時、衝突した

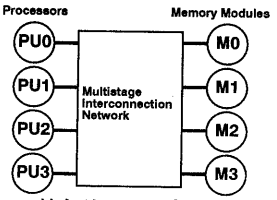

図 3: MIN 結合型マルチプロセッサの概念図

二つのパケットのうち片方のみが残り、もう片方は破棄され再び網の入口から送られる。各プロセッサはメモリモジュール M_0,M_1,M_2,M_3へ、それぞれ P_0,P_1,P_2,P_3の確率でアクセスを行なう。1 ステップ時間でプロセッサからメモリブロックへのアクセス要求とメモリモジュールからプロセッサへのアクノリッジが行なわれると仮定する。ここで結合網内でのパケットの衝突を無視すれば、図 4のような STMT ネットが得られる。

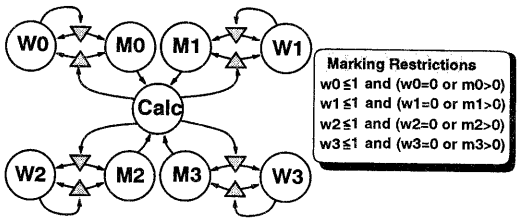

図 4: MIN 結合型マルチプロセッサの STMT ネット

Calc は共有メモリアクセスを伴わない処理を行なっている状態、$M_0,\cdots,M_3$はそれぞれのメモリモジュールにアクセスしている状態、$W_0,\cdots,W_3$はメモリモジュールが他のプロセッサに使われていたためアクセス待ちになっている状態をそれぞれ示す。ここで、MIN がブロッキング網であるため、アクセスするメモリブロックが異なるパケット同士でも衝突することをモデル化しなくてはならない。しかし一般にプロセッサの位置とアクセスするメモリブロックの位置の両方がわかっていないと、パケットの衝突を判定することができない。STMT ネットではプロセッサを区別することができないので、この判定を行なうことができず、モデル化が困難となる。

また、図 2に示したマルチプロセッサで、プロセッサによって異なる処理を行なっている場合、処理内容によってメモリアクセスを起こす確率 P_r,P_wが異なってくる。STMT ネットではトークン毎に異なる遷移確率を与えることができないため、この場合のモデル化は困難である。

以上の例以外にも、実際の並列システムでは、負荷分散, 同期, 等の点で各要素が部分的に異なった振舞いをする場合がある。そして、そのようなシステムをトークンの区別を行なわない STMT ネットで記述することは困難である。

3.2 Colored STMT ネットの概要

上に述べた STMT ネットの問題を解決するため、トークンに属性を持たせた、Colored STMT ネットを提案する。Colored STMT ネットは、トークンの属性を無視すれば STMT ネットと同一であり、マーキング制約, 条件付きパス, 分岐付きパス等の概念をそのまま採り入れている。

従ってユーザは、システムの基本的な動作をまずSTMTネットで記述し、後から細かな部分、すなわちトークン毎に異なる部分を書き加えることにより、複雑なシステムのモデルを容易に記述することができる。

STMTネットでは、システムの状態を各プレースに存在するトークン数の組として表し、これをマーキングと呼んだ。トークンが属性を持つと、システムの状態(マーキング)は、各プレースに存在するトークン数の組だけでは特定することができない。そこで、Colored STMTネットでは、個々の属性ごとにトークンのプレース毎の分布を表す組を作り、それらを全て集めてマーキングとする。そして、STMTネットと同様、このマーキングをもとに、制約や遷移条件を与える。

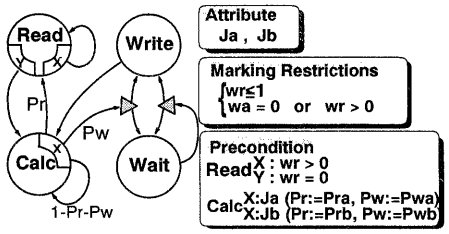

図5: プロセッサによってアクセスパターンが異なる場合のブロードキャストメモリに対するColored STMTネット

図5に、図2のマルチプロセッサにおいてプロセッサによりP_r, P_wが異なる場合をColored STMTネットで表した例を示す。この例は、二種類のジョブにそれぞれ幾つかプロセッサを割り当て並列に実行させた場合を想定している。属性J_aを持つトークンはジョブaを担当するプロセッサを表し、J_bを持つトークンはジョブbを担当するプロセッサを表す。図5のAttribute欄にシステム内に存在する属性名を記述する。P_r, P_wの値は、属性J_aのトークンに対しP_{ra}, P_{wa}、また属性J_bのトークンに対しP_{rb}, P_{wb}とする。$Calc$にいるトークンは自分の属性に対応した遷移確率を選ばなければならない。Colored STMTネットでは、条件付きパスによって以上の動作を記述することができる。条件付きパスでは、現在のマーキングの他に遷移をしようとするトークンの属性を条件として与えることができる。この例の場合、属性毎に専用のパスを、条件付きパスを利用して設定し、それぞれのパスにP_{ra}, P_{wa}もしくはP_{rb}, P_{wb}の遷移確率を与える。ただし、以上をそのまま状態遷移図に直すとパスの数が増え図が見にくくなるので、P_r, P_wの値が属性により変化することを図中のPreconditionの欄にまとめて記述する。

次に、図3で示したMIN結合型マルチプロセッサのColored STMTネットでの表現について述べる。ここで、MINは図6に示すオメガ網であるとする。図6において、各ステージにおいて異なる二つのパケットが衝突する条件は以下のようになる。(二つのパケットのプロセッサ番号とアクセスするメモリモジュール番号をそれぞれp_a, m_aおよびp_b, m_bとする)

$Stage1$: $p_a \bmod 2 = p_b \bmod 2$ かつ $m_a/2 = m_b/2$

$Stage2$: $m_a = m_b$

ただし'/'は整数除算を表す

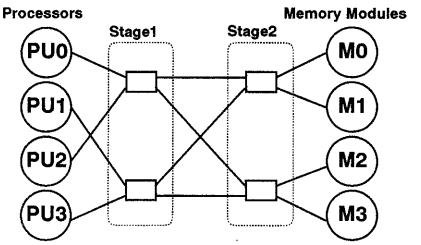

図6: オメガ網を結合網とするマルチプロセッサの概念図

以上の条件をマーキング制約として与えるためには、トークンを識別する属性を数値で指定できると便利である。Colored STMTネットでは、同じ属性を持つトークンには自動的に属性内でユニークな番号(属性内番号)が与えられる。そして、必要ならば属性に加え属性内番号によって各トークンを識別することができる。全プロセッサが属性PUに属するとし、属性内番号を用いてトークンを区別することにする。マーキング制約はSTMTネットと同様、トークンの数によって指定する。Colored STMTネットでは、あるプレースに存在するある属性をもったトークンの数を指定できる。MINでのパケットの衝突についてのマーキング制約を示す。

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

ここで、$M_i.PU(j)$はプレースM_iに存在するPUの属性を持つトークンのうち属性内番号がjであるトークンの数を表す。また$for(i = 1; i \leq 4)\{\cdots\}$は1から4を順に$i$に代入して$\{\cdots\}$の部分を評価しそれぞれをマーキング制約に加えることを意味する。図4のMarking Restriction欄に上記のマーキング制約を加えることによって、MIN結合型マルチプロセッサの動作の記述が完成する。

3.3 Colored STMTネットの定義

Colored STMTネットは次の6つ組で与えられる

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

ここで

$\mathbf{P_L}$はプレースの集合 : $\mathbf{P_L} = \{pl_1, pl_2, \cdots, pl_n\}$

$\mathbf{P_A}$は有向パスの集合 : $\mathbf{P_A} = \{pa_{11}, pa_{12}, \cdots, pa_{nn}\}$

$\mathbf{I_D}$は識別子の集合 : $\mathbf{I_D} = \{id_1, id_2, \cdots, id_m\}$

である。ただし、pa_{ij}はpl_iからpl_jへのパスを表す。各トークンは$\mathbf{I_D}$の中の一つの要素を識別子として持つ。識別子は属性のことではない。同じ属性を持つトークンは属性内番号により区別することができる。しかし、同じ識別子を持つトークンを区別することはできない。識別子id_iを持つトークンの数をn_iとすると

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

が成り立つ。トークンが識別子を持つため、システム内で許されているマーキングM_qは、プレースに存在しているトークンの数とその識別子の両方で表される。

システム内に識別子id_iのトークンのみが存在していると仮定した時のシステムのとり得るマーキング$M_q^{(i)}$は

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

で与えられる。$m_{qj}^{(i)}$はプレースpl_jに存在している識別子

id_iをもつトークンの数である。M_qは$M_q^{(i)}$を全ての識別子について集めたもので、

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

で表される。マーキング制約$\mathbf{R}$はシステム内で許されているM_qの集合として与えられる。

M_0はマーキングの初期値であり、

$M_0 = \{M_0^{(1)}, M_0^{(2)}, \cdots, M_0^{(m)}\} \in \mathbf{R}$

$M_0^{(i)} = \{m_{01}^{(i)}, m_{02}^{(i)}, \cdots, m_{0n}^{(i)}\}$

で与えられる。ここで

$n_i = \sum_{j=1}^{n} m_{0j}^{(i)}$

$N = \sum_{i=1}^{m}\sum_{j=1}^{n} m_{0j}^{(i)}$

が成り立つ。$\mathbf{P_R}$は各パスに与えられたトークンの移動確率の集合であり

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

で与えられる。STMTネットではトークンの遷移確率は現在のマーキングM_qの関数であるが、Colored STMTネットではM_qおよび遷移をするトークンの識別子の関数である。トークンは現在のマーキングと自分の識別子から定められる遷移確率に従って$\mathbf{R}$で与えられるマーキング制約のもとでプレース間を移動する。トークンの持つ識別子は静的であり、パスの通過やプレースへの到着に伴って変化することはない。

STMTネットは、識別子が一つしか存在しない$(m=1)$特殊なColored STMTネットである。従って、各識別子毎にプレース集合$\mathbf{P_L}$の複製を与え、そのプレース集合内では単一の識別子のトークンのみが存在するようにColored STMTネットを分解すると、もはや識別子は意味を持たなくなりSTMTネットになる。

このように各識別子に専用のプレース集合を与えることによって、$\{\mathbf{P_L}, \mathbf{P_A}, \mathbf{R}, M_0, \mathbf{P_R}, \mathbf{I_D}\}$で表されるColored STMTネットは、$\{\mathbf{P_L}', \mathbf{P_A}', \mathbf{R}', M_0', \mathbf{P_R}', \{id\}\}$で表される等価なSTMTネット(識別子がただ一つのColored STMTネット)に変換することができる。今$\mathbf{P_L}$で与えられるプレース集合をもとに、識別子id_iに対して$\mathbf{P_L}_i$なる専用プレース集合を与える。$\mathbf{P_L}_i$を

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

と表すと、全識別子分を集めた集合$\mathbf{P_L}'$は

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

である。システム全体でプレースの数は$n \times m$個となる。各プレースには定められた単一の識別子を持つトークンのみが存在できるので、マーキングM_q'は単に$\mathbf{P_L}'$の各プレースにいるトークンの数で次のように表される。

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

$\mathbf{P_L}_i$に含まれるプレースに存在するトークンの総数はn_iで一定であることから、実際に許されるマーキングの集合$\mathbf{R}^*$は

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

である。システム内で許されるマーキングの集合$\mathbf{R}'$は、$\mathbf{R}^*$の部分集合である。

トークンの持つ識別子は静的であるから、$j \neq l$とするとトークンがpl_{ij}からpl_{kl}に移動することはない。したがってトークンの移動確率の集合$\mathbf{P_R}'$を

$\mathbf{P_R}' = \{pr_{(11)(11)}, pr_{(11)(21)}, \cdots, pr_{(nm)(nm)}\}$

とすると

$pr_{(ij)(kl)} = 0 \qquad (j \neq l)$

である。ただし、$pr_{(ij)(kl)}$はpl_{ij}からpl_{kl}への遷移確率を表す。

4 Colored STMT ネットアナライザ

ユーザが並列システムの解析に容易にColored STMTネットを利用できるようにするために、専用記述言語Betaを用意した。Betaによりユーザは簡単かつ的確にシステムを記述することが出来る。BetaはSTMTネット記述言語Alphaの構造の多くを引き継いでおり、Alphaと同様、簡単であるが強力な記述力を備えている。これにより従来のSTMTネットのユーザは容易に言語Betaに親しむことができる。

Betaで記述されたColored STMTネットはプリプロセッサにより等価なSTMTネットに変換され、STMTネットアナライザ[4]により解析される。そして、平衡状態における各プレースのトークンの数の期待値を、各属性毎に(必要ならば属性内番号毎に)、求めることができる。

4.1 Colored STMT ネット記述言語 Beta

Betaでは、対象となるモデルを状態記述と遷移記述とに分けて記述する。状態記述部では、トークンの属性名、プレース名、マーキング制約などの記述を行う。遷移記述部では、プレース間のパスと、そのパスに与えられたトークン移動確率や前提条件などを記述する。

[状態記述部]

・属性宣言

%token {*attribute name*}({*max*})

Colored STMTネット中のトークンの属性名の宣言。*attribute*で示す属性を持ったトークンが*max*個存在することを表す。

・プレース宣言

%place {*place name*}

Colored STMTネット中のプレース名の宣言

・マーキング制約記述

%restrict {*expression*}

系のとるマーキングの満たすべき条件を設定する。この制約は、あるプレースに存在するある属性をもつトークンの数を表す変数@*place_name.attribute_name*と、C言語の演算子から構成される式*expression*で記述される。この変数には、次に示すような表記が許されている。

@$place_a$.$attribute_a$: $place_a$に存在し、属性$attribute_a$を持つトークンの数

@$place_a$.$attribute_a$(i_a,i_b,i_c) : $place_a$に存在し、属性$attribute_a$を持ちかつ属性内番号がi_a,i_b, もしくはi_cであるトークンの数

@$place_a$.$attribute_a$(i_a..i_b) : $place_a$に存在し、属性$attribute_a$を持ちかつ属性内番号が$i_a, i_a+1, \cdots, i_b$の範囲にあるトークンの数

MIN結合型マルチプロセッサの例で示した条件1はBetaでは以下のように表す。

```
1  #for( i=1 ; i<=4 ; i++ )
2  %restrict (@M0.PU(i % 2)+@M1.PU(i % 2)<=1)&&
             (@M2.PU(i % 2)+@M3.PU(i % 2)<=1)
3  #end
```

上の例で、#for と#end の間の文は $i=1$ から $i=4$ まで繰り返し定義される。

[遷移記述部]

プレースごとに、そこを始点プレースとするパスについて記述する。トークンが各パスを選択する確率は、重みとして各パスに与える。

$$\begin{array}{lcll} place_a & : & place_b & [weight_b] \\ & | & place_c & [weight_c] \\ & | & place_d & [weight_d] \\ & ; & & \end{array}$$

上の例では、$place_a$から $place_b$, $place_c$, $place_d$へパスがあり、それぞれの遷移確率が

$$\frac{weight_b}{weight_{total}}\ ,\ \frac{weight_c}{weight_{total}}\ ,\ \frac{weight_d}{weight_{total}}$$
$$(weight_{total} = weight_b + weight_c + weight_d)$$

である。トークンの属性によってパスを選択する場合の記述は以下のように行う。

$$\begin{array}{llcll} place_a & attribute_1 & : & place_b & [weight_b] \\ & & | & place_c & [weight_c] \\ & attribute_2 & : & place_d & [weight_d] \\ & & | & place_e & [weight_e] \\ & & ; & & \end{array}$$

上の例において、属性 $attribute_1$ を持つトークンは $place_a$ から $place_b$,$place_c$へそれぞれ

$$\frac{weight_b}{weight_b + weight_c}\ ,\ \frac{weight_c}{weight_b + weight_c}$$

の遷移確率で移動する。属性 $attribute_2$ を持つトークンも同様に $place_d$,$place_e$へ移動する。属性に属性内番号を指定するときの記述は以下のようになる。

$$\begin{array}{lll} place_a & attribute_1(i..j) & : place_b \\ & attribute_1(k..l) & : place_c \end{array}$$

属性 $attribute_1$ を持つトークンの中で属性内番号が $i,\cdots,j$ のものは $place_b$へ、また $k,\cdots,l$のものは $place_c$へそれぞれ確率 1 で遷移することを表している。

条件付きパスの記述は以下のように行う。

$$\begin{array}{llcll} place_a & \langle\langle condition_x\rangle\rangle & : & place_b & [weight_b] \\ & & | & place_c & [weight_c] \\ & \langle\langle condition_y\rangle\rangle & : & place_d & \\ & \langle\langle else\rangle\rangle & : & place_e & [weight_e] \\ & & | & place_f & [weight_f] \\ & & ; & & \end{array}$$

遷移条件 $condition_x$や $condition_y$は、%restrict 文の $expresstion$ と同様に式で記述する。この例では、$place_a$を始点とするパスが、$condition_x$が成立するとき、$condition_y$が成立するとき、ともに成立しないときの三つの場合について定義されている。

重み付けは、各条件ごとに評価されるので、$condition_x$ が成立するときの $place_a$から $place_b$への遷移確率は $weight_b/(weight_b + weight_c)$ となり、$condition_y$が成立するときは、$place_b$へ確率 1 で遷移する。

属性ごとに条件付きパスを与えるときの記述は以下のように行なう。

$$\begin{array}{llllll} place_a & attribute_1 & \langle\langle condition_x\rangle\rangle & : & place_b & \\ & & \langle\langle else\rangle\rangle & : & place_c & [weight_c] \\ & & & | & place_d & [weight_d] \\ & attribute_2 & \langle\langle condition_y\rangle\rangle & : & place_e & \\ & & \langle\langle else\rangle\rangle & : & place_h & \\ & & & ; & & \end{array}$$

この例では、$attribute_1$の属性を持つトークンについて $condition_x$が成立するときと成立しないときの二つの場合のパスの定義がなされている。また $attribute_2$の属性を持つトークンについて $condition_y$が成立するとき、としないときの二つの場合のパスの定義がなされている。属性内番号を変数として条件式中で使う場合は次のように記述する。

$$\begin{array}{llll} place_a & attribute_1(x := i..j) & \langle\langle condition\rangle\rangle & : place_b \\ & & \langle\langle else\rangle\rangle & : place_c \end{array}$$

上の例において、条件 $condition$ の式中に、x を含むことができる。$attribute_1$の属性をもち属性内番号が $i,\cdots,j$のトークンは、x に自分の属性内番号を代入したのち $condition$ を評価し、値が真ならば $place_b$へ、偽ならば $place_c$へそれぞれ遷移する。

分岐付きパスの記述は以下のように複数の終点を並べて記述する。

$$place_a \quad : \quad place_b\ ,\ place_c$$

これは、より、$place_b$と $place_c$へ移動するトークンの数が一意的に決まる場合の記述方法である。

5 Colored STMT ネットの適用例

Colored STMT ネットでは、従来の STMT ネットを利用した解析では得られなかったシステムの性能についての情報を得ることができる。

ここでは、過去に行なわれた STMT ネットによる実際の並列システムの解析例について、Colored STMT ネットを適用しシステムのより詳しい解析を行なった例を示す。

5.1 IEEE 標準バス Futurebus のアービトレーションプロトコル

IEEE Futurebus[6] は、マルチプロセッサシステムでの使用を前提とした標準バス規格である。Futurebus では、同時に複数のモジュールがバスの使用を要求した場合、アービトレーションプロトコルに従ってバスマスタが一つだけ選出される。バスマスタを決定するシーケンスをアービトレーションサイクルと呼ぶ。各モジュールにはシステム内でユニークなアービトレーション番号が割り当てられている。アービトレーションサイクルでは、このアービトレーション番号の値がもっとも大きいものがアービトレーションに勝ちバスマスタとなる。アービトレーション番号の低いモジュールが飢餓状態に陥ることを防ぐため、Futurebus ではフェアネスクラスというクラスをモジュールに対して用意し飢餓状態を回避している。これはフェアネス機構と呼ばれる。フェアネスクラスに属するモジュールには次のような制約が課せられる。

・バスマスタから退いたモジュールは、続けてアービトレーションサイクルへ参加することを禁止される。

この状態を禁止状態と呼ぶ。禁止状態を解除できるのは、バスの使用権を要求しているにもかかわらずまだバ

```
1   %token          Mdl(Max)
2   %place          FB
3   %place          C
4   %place          CM
5   %place          IB
6   %place          W

7   %restrict       (@CM == 1)

8   %%

9   FB      :       C       Prq
10          |       FB      1-Prq
11          ;

12  C   Mdl(Max)                                    :  C , CM
13      Mdl(i:=1..Max-1)<<@C.Mdl(i..Max)==0>> :  C , CM
14                      <<else>>                    :  C
15                                                  ;

16  IB <<@C ==0 && @W > 0>> :       FB
17      <<else>>            :       W       Prq
18                          |       IB      1-Prq
19                          ;

20  W <<@C ==0 && @W > 0>>  :       C
21    <<else>>             :       W
22                         ;

23  CM <<@C > 0 >>         :       IB
24      << else >>         :       CM
25                         ;
```

図 7: Beta による Futurebus のアービトレーションプロトコルの記述

スマスタになっていないモジュールが一つもいなくなったときに解除される。禁止状態を解除するサイクルをフェアネス解放サイクルと呼ぶ。この制約によってフェアネスクラスに属するモジュール内での飢餓状態は回避される。ただし、各プロセッサがバスを獲得できる機会は、アービトレーション番号が大きい方が多い。すなわち、フェアネス機構は弱公平性のみを保証している。

フェアネス機構のシステム性能に与える影響について調べるため、以上のアービトレーションプロトコルをSTMTネットでモデル化し解析が行なわた。その結果バスマスタ要求確率とフェアネス解放サイクイル発生確率との関係が明らかにされた [5]。ここではColored STMT ネットの特徴を生かし、Futurebus のアービトレーションの公平性について解析を行なう。

すべてのモジュールがフェアネスクラスに属する場合、各モジュールは次の 5 状態をとる。

current master(CM)	:現バスマスタ
free bystander(FB)	:禁止状態になく、バスの使用権を要求していない.
inhibited bystander(IB)	:禁止状態にあり、バスの使用権を要求していない.
competitor(C)	:禁止状態になく、バスの使用権を要求している.
witiholder(W)	:禁止状態にあり、バスの使用権を要求している.

以上の各状態をプレース、また、各モジュールを Mdl という属性を持つトークンとして表す。ここで属性内番号 i はアービトレーション番号に対応する。

図 7に Beta の記述を示す。Prqはモジュールがバスマスタ要求を出す確率である。また、Max はフェアネスクラスに属するするモジュールの数である。CM 状態にあるモジュールの数は常に 1 である。これはマーキング制約として与えられる。W 状態のモジュールが存在する時 C 状態にあるモジュールがなくなると、次のステップでフェアネス解放サイクルが生じ、IB,W 状態にあるモジュールはそれぞれ FB,C 状態に遷移する。これはプレース IB,W から FB,C に向かうパスが有効となる条件として与えられる。アービトレーションに勝って C 状態から CM 状態に遷移できるものはアービトレーション番号の大きいモジュールからである。これは、C から CM に向かうパスが有効となる条件として与えられる。

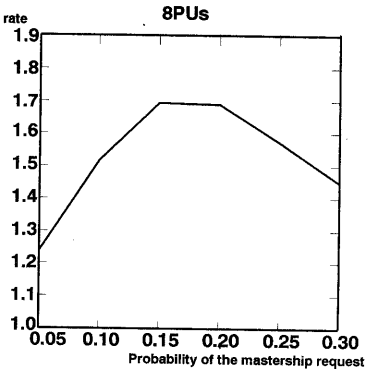

図 8: バスマスタ待ち状態である確率の最強モジュールに対する最弱モジュールの比

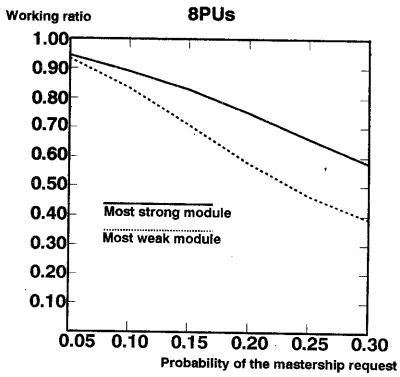

図 9: 最強モジュールと最弱モジュールのモジュール稼働率

モジュールの数を 8 とし、P_{rq}をパラメータとして解析を行なった結果を以下に述べる。以下の説明において、最も大きなアービトレーション番号を持つモジュールを最強モジュール、最も小さなアービトレーション番号を持つモジュールを最弱モジュールと呼ぶことにする。図 5.1に、モジュールがバスマスタ待ち状態である確率 (トークンがプレース C または W にある確率) について、最強モジュールに対する最弱モジュールの比を示す。P_{rq}が比較的小さい場合 P_{rq}の増加に従い、この比は大きくなる。これは、

C 状態にあるモジュールの数の期待値の増加により、最強モジュールと最弱モジュールの間で C 状態に留まる時間の差が開いてくるためである。グラフは $P_{rq} = 0.150$ 付近で最大となり、この時最弱モジュールは最強モジュールに比べ、1.7 倍程度高い確率でバスマスタ待ち状態であることがわかる。P_{rq}が更に大きくなると、バスマスタから退いたモジュールは直ちに状態 IB から W に移り、フェアネス解放を待つことになる。したがって、最強モジュールは状態 W で足並みを揃えられてしまい、最弱モジュールとの差が小さくなる。図 5.1に最強モジュールと最弱モジュールのモジュール稼働率 (バスマスタ待ち状態以外の状態にある確率) を示す。図 5.1と異なり、P_{rq}=0.200 を越えても最強モジュールと最弱モジュールの差はさほど縮まっていない。これは、P_{rq}が大きくなり、最強と最弱モジュールのバスマスタ待ち状態である確率の差が減少しても、バスマスタ待ち時間に費やす絶対的な時間が増え、この時間のプロセッサ稼働率に与える影響が増大するためである。

6 計算コストに関する考察

前述したように、全プレースの数が m, 識別子の数が n である Colored STMT ネットは、全プレースの数が $m \times n$ の STMT ネットに変換できる。したがって、各要素についての状態遷移図がそのままであっても、トークンに属性を付加することによって、システム全体の取り得る状態数は爆発的に増加する。マーキング数が非常に多くなると、初期マーキングと遷移確率行列から定常状態のシステムの挙動を得るまでに、多大な計算時間を要するようになる。また、遷移確率行列の計算にも多くの計算資源が必要となる。

従って、ユーザは、STMT ネットで表されたシステムを属性を使って更に詳しくモデル化するにあたり、その解析がどの程度の規模になるかあらかじめ予想をたて、その計算が可能かどうかを知っておく必要がある。

トークンに区別のない STMT ネットでは、システム内に存在するトークンの数を Nとすると、マーキング数は

$$M_{stmt} = {}_{N+m-1}\mathrm{C}_{m-1} = \frac{(N+m-1)!}{N! \cdot (m-1)!}$$

で表される。Colored STMT ネットですべてのトークンを区別する場合のマーキングの数は、

$$M_{c\_stmt} = m^N$$

である。ただし、M_{c_stmt}は各識別子をもつトークンの数が一定 (= 1) という条件により存在しないマーキングを削除した値である。これより同規模のシステムにおいて、各トークンを識別することにより、システムの取り得るマーキング数は M_{c_stmt}/M_{stmt}倍となる。

STMT ネットアナライザでは、モデル中で定義されたマーキング制約を用いて、存在しないマーキングを削除することにより、実際に扱うマーキング数 (有効マーキング数) を最小限に抑えている。従って、実際の計算量として実際に問題となるのは M_{c_stmt}でなく有効マーキング数である。

有効マーキング数を正確に求めることは簡単ではないが、属性を与える前の STMT ネットで表現されたモデルについての有効マーキング数が STMT ネットアナライザによりあらかじめ分かっていた場合、それに M_{c_stmt}/M_{stmt} の値を掛けることにより、ある程度有効マーキング数を予想することができる。これは、属性を付加する前と後で、全マーキング数に対する有効マーキング数の割合がほぼ同じであると仮定することにより得られる。

表 1: 有効マーキング数の比較

トークン数	属性なし	属性あり	予測値
2	4	8	6.6
4	20	256	178.6
6	56	6144	4166.6
8	120	131072	94697.0

表 1に Futurebus のアービトレーションプロトコルの解析 (図 7) において、トークンを区別しない場合 (C 状態からは条件無しに C もしくは CM 状態に遷移する) の有効マーキング数と、それから予想された有効マーキング数、実際の有効マーキング数の値を示す。表より、先の仮定に基づく予想値がだいたいの計算量を知る上で十分有効であることがわかる。

7 むすび

単純であるが幅広い用途を持つ理論解析モデルである STMT ネットを拡張し、トークンに属性を持たせた Colored STMT ネットの提案を行なった。Colored STMT ネットの導入により、STMT ネットでは記述が困難であった幾つかのシステムに対するモデル化が可能となった。実際に Colored STMT ネットを用いて、Futurebus のアービトレーションプロトコルについて解析を行なった。その結果、最強モジュールと最弱モジュールとの間の状態の違いが明らかになった。トークンを属性で区別することにより、システムの取り得るマーキング数は爆発的に増加する。このとき増加する計算量はある程度予測することができ、ユーザは計算可能な範囲で詳細なモデルを構築することができる。

8 謝辞

本研究を進めるにあたり、数々の御指導を頂いた慶應義塾大学計算機科学専攻天野研究室の皆様に感謝致します。

参考文献

[1] 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).

[2] 天野 英晴, 地川 淳二, 吉田 隆一, 相磯 秀夫: " マルチリード メモリを用いた並列計算機の性能解析", 信学論 **J67-D**, 9, pp.1028-1035 (1984-9)

[3] 鳥居 淳, 竹本 卓, 天野 英晴, 小椋 里: " バス結合型並列計算機の交信用メモリの評価", 情報処理学会論文誌 **33**, 3, pp.307-319 (1992-3)

[4] 木村哲郎, 山本欧, 天野英晴, 竹本 卓: " 並列システム解析モデル : STMT ネット", 電子情報通信学会論文誌,D-I,Vol.J75-D-I,No8, pp.654-663 (1992-8)

[5] 山本欧, 鳥居淳, 天野英晴: "IEEE 標準バス Futurebus のバスアービタの性能評価", 情報研報 91-ARC-14,pp.111-118 (1991-11)

[6] "IEEE Standard Backplane Bus Specification for Multiprocessor Architectures:Futurebus",IEEE Jun.,1988

盛光印刷所