マルチスレッド化目的コードを生成するLOTOSコンパイラの機能拡張

3Bb－2

後藤和裕[1]　安本慶一[2]　東野輝夫[1]　谷口健一[1]

[1] 大阪大学基礎工学部情報工学科　[2] 滋賀大学経済学部情報管理学科

1　はじめに

形式記述言語LOTOS[2]は，複数並行プロセス間でイベントの同期実行やデータ交換を行うための，マルチランデブと呼ばれる，高度なプリミティブを有する．

我々は，従来からマルチランデブを含むLOTOS仕様を，1つの計算機上でマルチスレッド機構を用いて効率良く実行するためのコンパイラを作成してきた[4]．本稿では，コンパイラの拡張として，ネットワーク上の異なる計算機間でのマルチランデブを含むLOTOS仕様を分散実行するための一つの方式を考案する．

2　LOTOSの概要

[オペレータ] LOTOSでは，システムの仕様をいくつかのサブプロセスからなるプロセスとして記述する．プロセスには，動作式として，システムの外部から観測可能な振る舞い，すなわち観測可能なイベントとそれらの時間的実行順序を指定する．イベントの実行順序の指定には，同期 ($|[a]|$)，並列 ($|||$)，選択 ($[]$)，割り込み ($[>$)，逐次実行 ($>>$) などのオペレータが用いられる．

[マルチランデブ] マルチランデブとは，2つ以上の並列に動作しているプロセスが，あるゲートを介して同時に1つのイベントを実行する機構のことである．

n 個のプロセスがマルチランデブによりイベントを同期実行できるのは，n 個のプロセス中の任意の2つのプロセスが，同一ゲートのイベントを実行可能であり，かつ，イベントの入出力値が次の生起条件を満たす時に限る．

p_i	p_j	同期条件	作用
$a!E_i$	$a!E_j$	$val(E_i) = val(E_j)$	値の照合
$a!E_i$	$a?x:t$	$val(E_i) \in domain(t)$	値の代入
$a?x:t$	$a?y:u$	$t = u$	値の生成

ある時点で前述のマルチランデブの生起条件を満たすようなイベントが複数ある場合には，それらのイベントの中から非決定的に1つのイベントを選択して生起しなければならない．

以後，あるゲート g を介してマルチランデブに参加するプロセスの集合を同期プロセスグループ (グループ) と呼び，$P(g)$ と記述する．

3　LOTOS仕様の実装環境

本稿では，図1のようなバス結合型ネットワークで接続された複数計算機上でLOTOS仕様を分散実行する方法を与える．

ここでは，(1) メッセージのブロードキャストによる1対多通信が可能，(2) ブロードキャストしたメッセージの到着順序はどのプロセスでも同じ，(3) メッセージは誤りなく有限時間内に到着，という3つの事項を満足するネットワークモデルを対象とする．

各計算機は1つのプロセスを実行するものとする．

図1:LOTOS仕様の実装環境

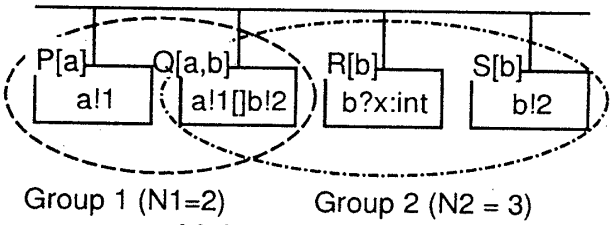
4　マルチランデブの分散アルゴリズム

従来より，複数の計算機上で並行動作しているプロセス間でマルチランデブを実現するためのアルゴリズムが提案されてきている[1, 3]．それらのアルゴリズムでは，プロセス間のオペレータには同期のみが指定されており，それぞれのグループで同期するプロセスの数は一定であると仮定している．

4.1　同期するプロセスの数が一定の場合

本節では，3章で提示したネットワークモデル上で，従来のアルゴリズムと同様な仮定のもとで，よりメッセージの交換回数が少ないアルゴリズムを提案する．

[各プロセスが知っている情報] あらかじめ，各グループに1から k までの番号を与えておく．また，プロセス $p_j (1 \leq j \leq n)$ は，(1) グループ番号 i の同期に参加するプロセスの個数 N_i(定数)，(2) 次に生起可能なイベントの集合 $G_{j,c}$ を知っているとする．



$spec := P[a] \,|[a]|\, Q[a,b] \,|[b]|\, R[b] \,|[b]|\, S[b]$

図2:同期するプロセスの数が一定の場合

[マルチランデブのアルゴリズム] 複数の計算機上で並行動作しているプロセス間でマルチランデブを実現する際には，各時点で同期実行可能なイベントを決定するための，(1) 同期条件を満たすグループを見つける，(2) 複数のグループで同期条件を満たす場合は，その中から1つのグループを非決定的に選択する，という2つの処理が，各イベント実行毎に必要になる．

以下では，上記を行うための各プロセス p_j の動作アルゴリズムを示す．

Step 0. [初期化] プロセス $p_1, \ldots, p_n$ が送信する同期要求のメッセージを保存するためのバッファを確保し，初期化する．各プロセスがブロードキャストしたメッセージはここに格納されるとする．

Step 1. [同期の要求] p_j は，$G_{j,c}$ を調べ，図3の形式のメッセージを作成し，自分自身を含む全てのプロセスに向かってブロードキャストする．p_j がイベントを実行するまで，再びメッセージをブロードキャストすることはない．

Extension of a LOTOS Compiler Generating Multi-threaded Object Codes

Kazuhiro GOTOH, Keiichi YASUMOTO,
Teruo HIGASHINO and Kenichi TANIGUCHI
Department of Information and Computer Sciences,
Faculty of Engineering Science, Osaka University, Japan
Department of Information Processing and Management,
Faculty of Economics, Shiga University, Japan

	ゲート名	入出力値	乱数
エントリ→	a	!1	133
	b	?x:int	573
		...	

図 3:メッセージの例

各メッセージには，$G_{j,c}$のイベントの個数分のエントリを持たせ，それぞれのエントリには，(1) 同期するゲート名，(2) イベントの入出力値，(3) 後述する実行グループの選択時に必要となる乱数，の 3 つを指定する．

Step 2. [同期条件の判定] p_jは，メッセージを受けるたびに，現在バッファに格納されている全メッセージについて，各イベント $a \in G_{j,c}$が同期条件を満たすかどうかを調べる．イベント a を含む全てのメッセージに対して，a の入出力値が同期条件を満たし，かつ，メッセージの個数が N_iである時，$P(g)$ で a が同期実行可能となる．

Step 3. [実行グループの選択] **Step 2.** で，実行可能なグループが複数あるような場合，次に述べる手順に従って 1 つのグループ $P(g)$ を非決定的に選択する．

各グループごとに，バッファのメッセージに付加された乱数の最大値を求め，乱数の最大値が最も大きいグループを選択する．バッファにあるメッセージはどのプロセスでも同じため，乱数の最大値は同じになる．もし，複数のグループで乱数の最大値が同じだった場合は，グループ番号の大きい方を選択する．

Step 4. [値の生成] 各 $p_i \in P(g)$ で生起するイベントの入力/出力の区別が全て入力であった時，最初にメッセージをブロードキャストしたプロセスが値を生成して，他のプロセスに値をブロードキャストする．

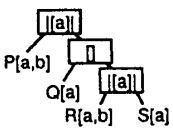
Step 5. [イベントの実行] 各 $p_i \in P(g)$ がブロードキャストしたメッセージをバッファから削除する．

$p_j \in P(g)$ の場合，gで生起されるイベントを実行し，次に p_jで生起可能なイベントの集合 $G'_{j,c}$に対して **Step 1.** からの処理を繰り返す．それ以外の場合は，再び **Step 2.** からやり直す．

4.2 同期するプロセスの数が可変の場合

本節では，プロセス間に選択などが指定されており，同期するプロセスの数が変化するような場合に対処できるよう，4.1節のアルゴリズムを拡張する．これは，文献 [1, 3] では扱えなかったクラスである．

[オペレータ情報] LOTOS の動作式を，プロセスを葉とする構文木で表した時，根から各プロセス p_jへ至る途中のオペレータの種類と接続位置 (右か左か) の対の順序付きリストを，p_jのオペレータ情報と定義する．



P	a	−L					
Q	a	−R,	[]	−L			
R	a	−R,	[]	−R,	a	−L	
S	a	−R,	[]	−R,	a	−R	

図 4:オペレータ情報の例

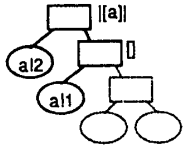


図 5:制御領域

	ゲート名	入出力値	乱数
	a	!1	133
	b	?x:int	573
		...	

オペレータ情報

a	−R,	[]	−R, ...

図 6:メッセージの例

[制御領域] プロセス間のオペレータを実現するための局所的作業領域として，前述の構文木からプロセスを除いた部分木の構造を持つ制御領域を各 p_jに用意する．

各プロセス p_jの動作アルゴリズムを次に示す．

Step 1. [同期の要求] p_jは，$G_{j,c}$を調べ，オペレータ情報を含むメッセージを作成し (図 6)，自分自身を含む全てのプロセスに向かってブロードキャストする．

Step 2. [同期条件の判定] p_jは，メッセージを受けるたびに，メッセージ中のオペレータ情報から，制御領域を構築し，$G_{j,c}$を葉の部分に接続する．

そして，文献 [4] で提案しているアルゴリズムを用いて，実行可能なイベントの集合 G_{exec}を計算する．

Step 3. [実行グループの選択] 4.1節で提案したアルゴリズムと同じ手法で，G_{exec}から 1 つのイベント gを選択する．

Step 4. [値の生成] 4.1節で提案したアルゴリズムと同様，必要に応じて値を生成して，他のプロセスにブロードキャストする．

Step 5. [イベントの実行] 各 $p_i \in P(g)$ がブロードキャストした $G_{i,c}$を制御領域から取り除き，根から各 $p_i \in P(g)$ へ至る途中のオペレータのうち，選択 ([]), 割り込み ([>) のノードに，どちらが選択されたのかを書き込む．次に，$P(g)$ と選択や割り込みの関係にあり，実行不可能になったプロセスの集合 P_{del}を求め，各 $p_l \in P_{del}$がブロードキャストした $G_{l,c}$も制御領域から取り除く．

$p_j \in P(g)$ の場合，gで生起されるイベントを実行し，次に p_jで生起可能なイベントの集合 $G'_{j,c}$に対して **Step 1.** からの処理を繰り返す．また，$p_j \in P_{del}$の場合，p_jを消滅させる．それ以外の場合は，再び **Step 2.** からやり直す．

[新たなプロセスの配置] $P := a;(Q|[a]|R)$ のように，イベントを実行した後に新たに生成されるプロセスを別の計算機に配置する場合，新しくオペレータ情報を作成しなければならない．これは，現在のプロセスのオペレータ情報の最後に，新たに生成されるプロセス間のオペレータ情報を追加することによって新たなオペレータ情報を得ることができる．

5 おわりに

本稿では，バス結合型ネットワークで接続された複数計算機上で LOTOS 仕様を分散実行するための，メッセージのブロードキャスト機能による 1 対多通信を用いたマルチランデブのアルゴリズムを提案した．

文献 [1, 3] で提案されているアルゴリズムでは，同期実行するイベントを決定する際，各プロセスで選択実行される全てのイベントについて，対応する同期グループの同期条件を判定する必要があったが，提案するアルゴリズムでは，1 つのグループでの同期条件が成立した時点でイベントを実行できるため，実行効率が良くなることが見込まれる．

今後，アルゴリズムの実装および LOTOS コンパイラへの組み込みを行い，評価・実験を行う予定である．

参考文献

[1] Z. Cheng, T. Huang, and N. Shiratori: "A New Distributed Algorithm for Implementation of LOTOS Multi-Rendezvous", *Proc. of the 7th Formal Description Techniques (FORTE'94)*, pp. 483-494 (1994).

[2] ISO : LOTOS: A Formal Description Technique based on the Temporal Ordering of Observational Behaviour, *ISO 8807* (1989).

[3] K. Naik: "Distributed Implementation of Multi-rendezvous in LOTOS Using the Orthogonal Communication Structure in Linda", *Proc. of the 15th Int. Conf. on Distributed Computing Systems (ICDCS'95)*, pp. 518-525 (1995).

[4] Yasumoto, K., Higashino, T., Abe, K., Matsuura, T. and Taniguchi, K.: A LOTOS Compiler Generating Multi-threaded Object Codes, *Proc. of the 8th Formal Description Techniques (FORTE'95)*, pp. 271-286 (1995).