周期スレッドを用いた実時間LOTOSコンパイラの設計

2J－3

辰本比呂記[1]　安本慶一[2]　安倍広多[3]　東野輝夫[1]　松浦敏雄[3]　谷口健一[1]

[1]大阪大学基礎工学部情報科学科　[2]滋賀大学経済学部情報管理学科　[3]大阪市立大学学術情報総合センター

1　はじめに

ソフトリアルタイム環境において，ビデオ・オン・デマンドや遠隔会議システムなどの分散マルチメディアシステムで一定の品質のサービスを提供するためには，サービスに必要なシステム資源（ネットワーク使用帯域やCPU時間等）の調整（QoS制御）が必要になる．これらQoS制御が必要なシステム（以下QoSシステムと呼ぶ）をFDT(形式仕様記述言語)により記述し，実装するための研究が行われている[2, 3, 4]．文献[4]では，仕様記述言語Estelleに時相論理を拡張した言語を用いて記述したQoSシステム仕様を，実時間OS上のマルチスレッド機構を用いて実装する方法を提案しているが，複数並行モジュール間のインタラクションを考慮したQoS制御を実現していない．LOTOS[1]は，並行，割込みおよびマルチランデブ（複数並行モジュール間で，指定された条件が成立する時のみ動的に同期通信を行う機構）などの高度な同期機構を持つ形式記述言語の一つであり，各イベントの実行に時間制約が指定できるよう拡張された言語（以下，時間付きLOTOS）も幾つか提案されている[2, 3]．これらの時間付きLOTOSではマルチランデブ機構と時間制約を組み合わせて利用することによりQoSシステムを簡潔に記述できるが，その実装法は未だ研究中である．

本稿では，ある時間付きLOTOSで記述されたQoSシステム仕様を周期スレッド機構[6]を用いてマルチスレッド化された目的コードに変換するコンパイラの設計法を提案する．周期スレッドでは，他のアプリケーションが使用するCPU時間やI/O処理などの要因により各周期に割当てられた処理の実行終了を保証できない．提案する手法では，複数スレッド間の実行優先度の変更，および，各周期におけるデッドラインとデッドライン超過時の代替処理等を指定することで，仕様に書かれた時間制約を最大限満たすよう動的に調整を行う．

2　LOTOSの時間拡張

本節ではLOTOSの概要及び我々が追加した時間制約記述のための構文を含んだ実時間拡張LOTOSについて述べる．以下，前者を従来LOTOS，後者を時間付きLOTOSと呼ぶ．

従来LOTOSでは，システムの仕様を同期（||），並列（|||），選択（[]），割り込み（[>），逐次実行（>>）などの関係が指定された，いくつかのプロセスとして記述する．各プロセスは処理の最小単位であるイベントの実行系列として定義される．

従来LOTOSは，マルチランデブと呼ばれる，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を有する．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個の全プロセスが，指定された同じゲートで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

時間付きLOTOSは，従来LOTOSに時間指定のための構文を追加したものであり，従来LOTOSと同様，選択・並列・割込み・マルチランデブなどの機構を有する．時間付きLOTOSは，次のような構文で定義される．

```
B       ::= B>>B | B[>B | B||B | B|[gatelist]|B | B|||B
        | B[]B | Event;B | Exit | stop | Process | wait(d)
Event   ::= GateID valdec [guard]{start,end}
Process::= ProcessID[gatelist](parameterlist){start,end}
Exit    ::= exit(parameterlist){start,end}
```

以下は新たに追加された構文とその意味である．

構文	意味
$e\{t1,t2\}$	イベントeは$t1 \le t \le t2$を満たす時刻tに実行できる
$exit\{t1,t2\}$	プロセスは$t1 \le t \le t2$を満たす時刻tに終了できる
$P\{t1,t2\}$	プロセスPは$t1 \le t \le t2$を満たす時刻tに呼び出されることができる
$wait(d)$	d単位時間待つ

時刻はすべてプロセスが呼び出された時点からの相対時刻で指定する．また，時間指定が満たされない場合には，イベント・プロセス呼出しは実行されないものとする．

3　QoSシステムの記述

本節では，時間付きLOTOSを用いたQoSシステムの一つの記述法を提案する．まず，どのような動作をどのような順序で行なうのかのみを記述した，時間指定を含まないシステム動作仕様と，システムの守るべき時間制約を記述した，時間指定を含むQoSシナリオをそれぞれ別プロセスとして記述し，これらを同期させることでQoSシステム全体を記述する．このように主動作仕様を補助する形の制約指向スタイルで記述することでシナリオの変更・保守が容易となる．

QoSシナリオは，一般にはシステムの提供するサービスが満たすべき品質（転送レート，最大遅延時間等）に基づく時間制約を記述したものであるが，時間制約が満たされない場合には動的にQoS調整するといった代替処理も合わせて記述する必要がある．

下の仕様は音声と動画の再生を行なうあるQoSシステムの記述例である．音声・動画データを読込むためのReadAudio(), ReadVideo(), 各データdataを画質vqで再生するOutputVideo(data:video,vq:quality), OutputAudio(data:audio,vq:quality)といった機構が予め用意されているものとする．

プロセスMainには音声・動画データをそれぞれ読み込み再生するというシステムの動作仕様を記述し，プロセスQoSには音声・動画の再生にかかった時間に基づいてQoSを調整するといった代替処理を記述している．この例では何らかの理由で0.03秒おきにフレームの更新ができない場合には画質を半分に落とし，混雑から回復後徐々に画質を戻す．

```
process System[vbuf,abuf,hard] : exit :=
  Main[vbuf,abuf,hard]
|[hard]|
  QoS[hard](vq0)
endproc

(* メイン動作 *)
process Main[vbuf,abuf,hard] : exit :=
  (vbuf?vd:video[vd = ReadVideo()]; exit
||| abuf?ad:audio[ad = ReadAudio()]; exit)
>>
  hard?vq:quality!OutputVideo(vd,vq)
                !OutputAudio(ad,vq);
  Main[vbuf,abuf,hard]
```

Design of timed-LOTOS Compiler with Periodic-Thread
Hiroki TATSUMOTO[1], Keiichi YASUMOTO[2], Kota ABE[3], Teruo HIGASHINO[1], Toshio MATSUURA[3] and Kenichi TANIGUCHI[1]
[1]Dept. Information and Computer Sciences, Osaka University,
[2]Faculty of Economics, Shiga University,
[3]Media Center, Osaka City University

```
endproc

(* QoS シナリオ *)
process QoS[hard](vq:quality):noexit :=
  ( hard!vq?vd:video?ad:audio;
    ( QoS[hard](vq+1){0,0.02}
  □  QoS[hard](vq){0.03,0.03}
    )
  ) [> QoS[hard](vq/2){0.03, ∞}
endproc
```

4 周期スレッドを用いた実装

我々が実装した周期スレッド機構 [6] では，各スレッドはユニークな周期を持ち，1 周期に 1 回，周期開始時刻から CPU が割り当てられる．また，同様にデッドライン (CPU 使用限界時刻) を指定できる．周期開始時刻に CPU が割り当てられない場合やデッドラインまでに処理が終了しない場合に行う処理はデッドラインハンドラとして指定する．周期，デッドラインは実行中に動的に変更できる．本節では周期スレッドを用いた実装の基本方針を述べる．

時間付き LOTOS 仕様を周期スレッドにマッピングする際に，コンパイラは仕様を解析し以下の事項を決定する．

(1) 仕様のどの部分を 1 つのスレッドに割り当てるか

基本的には 1 つのプロセスを 1 つのスレッドに割り当てる．ただし並列実行されるべき関係にある部分動作式を含む場合は，再帰的にこれらの動作式をそれぞれスレッドに割り当てる．また，自プロセスの呼び出しは可能であれば繰り返しとして実現する．詳細は論文 [5] 参照．

(2) スレッドの周期

プロセス中のイベント系列や自プロセス呼び出しに対する時間指定から対応スレッドの周期を決定する．基本方針としてプロセスにおける 1 ループをスレッドの 1 周期で処理する．例えば，P := B >> P{T, T} の形のプロセスの場合，プロセス P の呼び出しの条件から周期を T に決定する．

P:= (B1 [] B2) >> P のような形の動作式の場合，部分動作式 B1 と B2 の実行時間の違いによりスレッドの周期を一意に決定することができないが，周期は実行頻度の高い系列に合わせ，他は例外処理として割り当て CPU 時間を調整する．

(3) スレッドのデッドライン

使用可能な資源の総計，1 ループのイベント系列の処理時間に基づいて，各スレッドに割当てるべき CPU 時間を決定し，デッドラインを定める．初期値は，仕様を解析しある程度予測するか，あるいは初期値には適当な値を割り当て，実行履歴から CPU 時間の再配分を行い自動的に修正する．

```
P := B >> P{T,T} [> Q{T,}
周期Tの周期スレッド    デッドラインハンドラ
```

上図は周期的なプロセスの典型的な記述と周期スレッドの対応例である．この場合プロセス P を周期 T の周期スレッドに割り当て，Q はそのデッドラインハンドラに割り当てる．

次に，時間制約を満足させるための QoS 制御の基本方針について説明する．一般に，動作仕様にはシステムの守るべき時間指定のみを記述するが，コンパイラは時間制約を満たすように動作するための制御ルーチンを生成し，目的コードに組み込む．このコードは以下のような動作を行なう．

まず，各イベント系列の CPU 消費時間および I/O 処理時間が前もって与えられるものとする．これらは過去のループ実行にかかった時間の統計から求まる．前述の方法で各スレッドは割り当てられる資源の割合を決定する．ただし，ソフトリアルタイム環境では資源の予約が明確にできないので，経過時間とイベント系列の実行状況から資源の割当てを動的に調節する必要がある．具体的には，各イベントの平均実行時刻と実際に実行された時刻を比較することでスレッドごとの CPU 時間消費率が分かるので，余裕のないスレッドのデッドラインを延長して割り当てる CPU 時間を増やす．

また，プロセスが増える環境ではアドミッション・コントロール (他プロセスの時間制約に影響を与えずにプロセスを追加できるか) についても考慮する必要がある．これは CPU 時間の余裕と追加プロセスの必要 CPU 時間から判断する．

最後に，時間付き LOTOS では従来 LOTOS におけるマルチランデブに加えて，時間制約も考慮したマルチランデブの指定が可能である．これにより時間指定を含むプロセス間での同期の記述が簡潔になる．時間制約付きマルチランデブでは 2 節で述べた同期の成立条件に加えて，同期する全プロセス間で実行可能時刻の範囲に共通部分がある必要がある．各スレッドは以下の手順で実行可能性の判定を行う．ただし，$e\{t1, t2\}$ の時 $t1$ を $start(e)$，$t2$ を $end(e)$ で表す．

1. 時間指定に無関係に実行可能なイベントの集合 E を求める (論文 [5] 参照)．
2. $E' = \{e | e \in E, start(e) \leq$ 現在時刻 $\leq end(e)\}$ を求める．
3. $E' \neq \phi$の時，E'から一つ選んで実行する．
   $E' = \phi$の時，現在時刻 $< start(e')$ かつ $start(e')$ が最小の e'を E から一つ選択し，$start(e') -$ 現在時刻 sleep 後実行．
   $E' = \phi$かつ，$\forall e \in E(end(e) <$ 現在時刻) の時は，スレッドを終了する．
4. 上記 1. から繰り返し．

5 実験と考察

実際に周期スレッドを用いて時間付き LOTOS 仕様を実現する準備段階として，(非周期) マルチスレッド化目的コードにループ開始時刻と時間指定の比較から時間的実行可能性を調べる機構を組み込んで動作実験を行なった．

具体的には，同一周期を持つ 5 つの並行プロセスを動作させた．各プロセスは 1 つのイベントを繰り返し生起させ，ある値を端末に出力するというものである．周期を 0.1 秒程度以下に設定すると非周期スレッドの場合 I/O 処理にかかりきりになり，システム全体としては CPU 時間に 50%近い余裕があるにもかかわらず周期的に CPU が割り当てられないスレッドが存在し，数秒でシステムが停止してしまう．これらは周期スレッドを用いることでより低いレベルでの時間制御を行ない，また，アプリケーションレベルでの QoS 制御を行なうことで改善されることが予想される．

6 おわりに

本稿では，従来 LOTOS に時間制約を記述するための構文を加えた時間付き LOTOS の提案，及び時間付き LOTOS で記述されたシステム仕様を実時間マルチスレッド化目的コードとして実現する手法の提案を行った．今後，周期スレッドを用いて本手法を実装してコンパイラへの組み込み，最終的には実際に時間付き LOTOS で記述された QoS システム仕様から得られる目的コードの性能評価を行う予定である．

参考文献

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

[2] L. Leonard and G. Leduc: An Introduction to ET-LOTOS for the description of time sensitive systems, Computer Networks and ISDN systems, 29(3): 271-292 (1997).

[3] J.-P. Courtiat and R. C. de Oliveira: RT-LOTOS and its application to multimedia protocol specification and validation, Proc. of IEEE Int. Conf. on multimedia Networking: 31-45 (1995).

[4] S. Fischer: Implementation of multimedia systems based on a realtime extension of Estelle, Formal Description Techniques IX: 310-326 (1996).

[5] 安本，安倍，後藤，東野，松浦，谷口：マルチスレッド化目的コードを生成する LOTOS コンパイラの実現，情処論誌，39(2) (to appear in 1998).

[6] 安倍，松浦，安本，東野，谷口：UNIX 上で周期スレッドを実現するユーザレベルスレッドライブラリの実現法，信学技報 CPSY-97-24: 49-54 (1997).