並列論理シミュレーション向きタイムワープ機構の効率的な実現手法

松本幸則[†] 瀧 和男[††]

実現容易性と処理効率をともに考慮した論理シミュレーション向きタイムワープ機構の実現手法を提案する．タイムワープ機構は，並列論理シミュレーションに代表される並列イベントシミュレーションの時刻管理機構として用いられる．タイムワープ機構を用いたシミュレーションの効率を高めるためには，ロールバックオーバヘッドを低減する工夫が重要である．さらに，この工夫は容易に実現可能であることが望ましい．本稿で提案するRCTW (Rollback-Counting Time Warp)，メッセージへのロールバック番号付けというきわめて簡単な操作を導入したものであり，1: メッセージ取消判断を行うタイミングを工夫することにより，プログラムのデータ構造が単純化される，2: プロセス間の通信においてメッセージ送受信の順序が保存されない環境で適用できるアンチメッセージ削減機構が実現できる，という2つの特長を有する．

An Efficient Implementation Method of the Time Warp Mechanism for Parallel Logic Simulation

YUKINORI MATSUMOTO † and KAZUO TAKI††

This paper presents a new method of implementing the Time Warp mechanism for logic simulation — RCTW (Rollback-Counting Time Warp) — which takes account of both the easiness of implementation and efficiency. The Time Warp mechanism is used for time-keeping in parallel discrete event simulation. Since the mechanism includes the rollback overhead, some device to reduce the overhead is required, while it is important that the device can be widely applied but its implementation should not be too complicated. RCTW employs a simple operation : rollback counting. This leads to advantageous features as follows. 1: By deciding messages to be canceled when they are being evaluated, efficient execution is realized with a very simple data structures of the scheduler and processes. 2: The number of antimessages can be reduced even in case that the message sending order is not preserved when they are received.

1. はじめに

論理シミュレーションはLSI設計工程の1つであり，回路論理の検証，信号伝播遅延の検証などを目的とする．論理シミュレーション工程は多大な計算時間を必要とすることから，その高速化が望まれている．近年，マルチプロセッサ型の超並列スーパコンピュータが数多く発表されてきており，これらの上で動作する高速な並列論理シミュレータへの期待は大きい．

並列論理シミュレーションは，並列イベントシミュレーションの問題として扱うことができる．並列イベントシミュレーションにおいては，イベント処理順序を制御する時刻管理機構が処理速度に大きく影響する．時刻管理機構は，集中時刻管理機構と分散時刻管理機構に大別され[5]，分散時刻管理機構はさらに保守的手法 (Chandy-Misraの方法)[13]と楽観的手法 (タイムワープ機構)[8]に分けられる．これらのうち，タイムワープ機構は，逐次ボトルネックが存在しない，およびデッドロックが発生しないという特長を持つことから有望視されている．

タイムワープ機構には，ロールバック処理というオーバヘッド要因がある．このため，処理の効率化を目指して，そのオーバヘッド低減の研究が行われてきた．具体的には，遅延取消 (lazy cancellation)[7),9)]，遅延再評価 (lazy reevaluation)[5]，メッセージ一括取消[6]，アンチメッセージ数削減[11]などの手法が提案されている．また，局所的なスケジューリング[2]や，大域的なスケジューリング[1),10),15)]の研究も行われて

† 三洋電機株式会社 東京情報通信研究所 新情報処理研究室
Advanced Information Processing Lab., Tokyo Information & Communication Research Center, SANYO Electric Co., Ltd.
†† 神戸大学 工学部 情報知能工学科
Department of Computer and Systems Engineering, Kobe University

いる．

しかしながら，これらの研究においては，実現容易性の視点から議論されることはなかった．一般に，逐次プログラムに比べ，並列プログラムの開発および保守コストは飛躍的に高い．したがって，ソフトウェア工学的な立場からは，タイムワープ機構を効率化するアルゴリズムが，容易にプログラムとして実現でき，かつ保守も容易であることがきわめて重要と考えられる☆．

以上の背景から，本稿では，実現の容易性および処理効率をともに考慮したRCTW（Rollback-Counting Time Warp）を提案する．RCTWは，主たる応用分野を論理シミュレーションとし，局所スケジューリングを考慮した際の実現を容易にするものである．具体的な工夫としては，メッセージに対するロールバック発生番号付けとその比較という簡単な操作を導入している．この操作は，メッセージ取消判断を，アンチメッセージ受信時でなくイベントメッセージ処理時に行えるようにするためのものである．上記操作の導入により，メッセージ処理のスケジューリングに関するデータ構造がきわめて単純化されている．また，RCTWでは，暗黙のうちにアンチメッセージ削減機構が実現されていることから，高い処理効率が同時に期待できる．しかも，このアンチメッセージ削減機構は，プロセス間の通信においてメッセージ送受信の順序が保存されない場合でも正しく機能する汎用性の高いものである．

以下，2章でタイムワープ機構の概要，および論理シミュレーションを適用対象としたアンチメッセージ数削減の技法について述べる．3章でスケジューリングにかかわる問題点を指摘し，4章でRCTWの詳細を説明する．5章ではRCTWの効果をデータ構造および処理効率の面から議論し，最後に6章で本稿をまとめる．

2. タイムワープ機構と論理シミュレーション

2.1 タイムワープ機構の概要

イベントシミュレーションは，複数のプロセスがメッセージ通信を行う形にモデル化できる．メッセージはイベント情報を持つとともに，イベント生起時刻が記されている．各プロセスは状態を持ち，メッセージ受信によってその状態を変化させる．たとえば論理シミュレーションでは，信号値の変化をイベントに，ゲートやフリップフロップをプロセスに対応させるのが一般的である．

イベントシミュレーションを正しく実行するためには，各プロセスにおいて，時刻順にメッセージを処理しなければならない．タイムワープ機構では，各プロセスが局所時刻（Local Virtual Time，以下LVTと略記）を持ち，これに基づいてメッセージ処理の順序を管理する．LVTは処理メッセージの時刻に伴って更新される．LVTが単調増加している間は，時刻順にメッセージ処理されていることになり，シミュレーションが正しく進んでいると考える．しかしながら，実際の並列処理環境ではメッセージを時刻順に処理できない，すなわちLVTが減少する場合が存在する．このような状況に備え，プロセスは入力メッセージ，出力メッセージおよびプロセス状態の履歴を保存しながら処理を進める．そして，メッセージ処理順序の誤りを発見したところで履歴を巻き戻し（ロールバック），処理のやりなおしをする．また，その時点で誤って送信したことが判明したメッセージに対しては，メッセージを取り消す役割を持つアンチメッセージ（取消対象メッセージと同じ時刻を持つ）を送信する．以上の処理によりシミュレーション結果の正当性が保証される．

このほか，メモリ管理・終了検出などのため，ときどき大域的な時刻（Global Virtual Time，以下GVTと略記）を求める必要がある．詳細については文献8)を参照されたい．

2.2 論理シミュレーション適用時の効率化技法

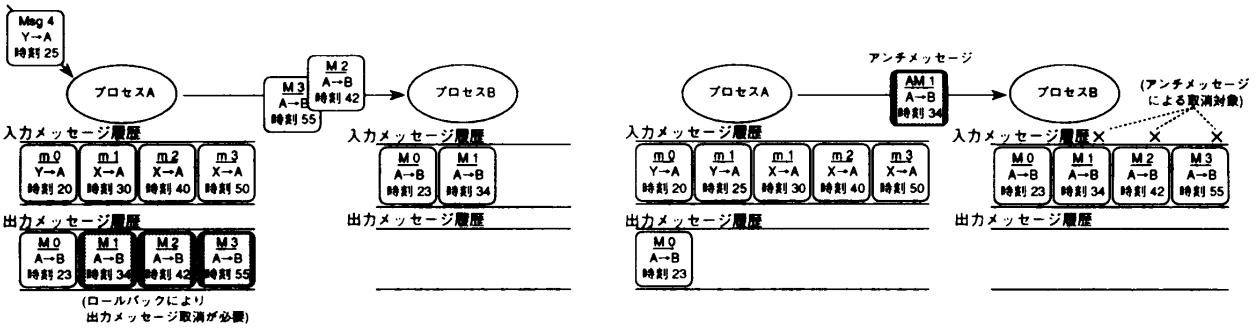
タイムワープ機構を現実の問題に適用して効率的に動作させるためには，ロールバック処理のオーバヘッドは小さい方が望ましい．ロールバックオーバヘッドを低減する手法の1つとしてアンチメッセージ削減機構が提案されており，実際に論理シミュレーションに適用した場合の有効性が報告されている[11]．この手法は，1つのアンチメッセージ送信で複数のメッセージ（本来のイベントに対応したメッセージ：以後イベントメッセージと呼ぶ）を一括して取り消すことにより，アンチメッセージ数を削減するものである．以下に簡単に説明する．

いま，対象問題に，

- 1つのプロセスにおいて，時刻の異なる2つの入力メッセージ m_{in1}，m_{in2}，およびそれぞれに対応した出力メッセージ m_{out1}，m_{out2} があるとき，m_{in1} の時刻が m_{in2} の時刻より小さければ，必ず m_{out1} の時刻が m_{out2} の時刻より小さい．

という性質があると仮定する．これを「優先イベントなしの性質」と呼ぶことにする．この性質は割り当て

☆ JPL（Jet Propulsion Lab.）のシステムでは，遅延再評価を組み込んだが，複雑さゆえに保守上の問題が発生し，後に除去されたことが文献5)に紹介されている．



(a) プロセスAにメッセージM4が遅れて到着し，ロールバック発生　(b) アンチメッセージAM1により，M1，M2，M3が取り消される

図1 アンチメッセージ削減手法
Fig. 1 The basic idea of antimessage reduction.

遅延モデルの論理シミュレーションなど多くの問題において成立する性質である．

優先イベントなしの性質を前提に，**図1**(a)の場合を考える．図1(a)は，プロセス A から B へ，複数のイベントメッセージ（M_0, M_1, M_2, M_3）が送信された後，A にメッセージが遅れて到着した状況を示している．その結果，ロールバックが発生し，B へ送信済みのメッセージのうち，M_1, M_2, M_3 を取り消す必要がある．

オリジナルのタイムワープ機構[8]では，メッセージ M が取り消し対象となるのは，M 送信後に，M と同じ時刻を持つアンチメッセージ AM が送信された場合となる．したがって，この場合，M_1, M_2, M_3 それぞれに対応したアンチメッセージを送信しなければならない．

しかし，優先イベントなしの性質があれば，次の定理がいえる．

定理1 メッセージ M が取消対象である必要十分条件は，「M 以後に送信され，かつ M 以下の時刻を持つアンチメッセージが M と同じ送信元から送信されていること」である．(証明略)

さらに，実行環境に，

- プロセス間の通信においてメッセージ追い越しがない（通信路のFIFO性と呼ぶ）

という性質があれば，以下に示すようなアンチメッセージ送信数削減ができる．

A では，取り消すべきイベントメッセージのうち，最小時刻を持つイベントメッセージ（この場合は M_1）に対応したアンチメッセージ AM_1 のみを送る．受信側の B においては，AM_1 受信時に，取消対象のメッセージすべてが受信済みであることが保証される（通信路のFIFO性より）．また，このとき，入力メッセージの履歴中，送信元が A で，かつ AM_1 の時刻以上のものが取り消されるべきイベントメッセージとなる（定理1より）．したがって，アンチメッセージ受信時に，これらを一括消去すればよい（図1(b)）．

3. スケジューリングに伴う問題点

3.1 論理シミュレーションにおけるスケジューリング

一般に並列論理シミュレーションでは，各プロセッサに複数ゲートからなる部分回路が割り当てられる．そして各ゲートはプロセスとして扱われる☆．この場合，メッセージ処理可能なゲートプロセスが1プロセッサ内に複数存在することから，各プロセッサにスケジューラを配置する必要がある．スケジューラの効率はシミュレーションの処理速度に大きく影響するため，スケジューラ導入に際しては，

- スケジューリング戦略
- スケジューリングキューの構造
- スケジューリング対象

の3点を考慮し，高い効率を実現する必要がある．

スケジューリング戦略については，ラウンドロビンや最小時刻優先などがある[2]．通常は，最小時刻優先戦略が適切と考えられている[1),8),12)]．

一方，スケジューリングキュー構造については，逐次シミュレーションで用いられるイベントキュー構造が参考になる☆☆．論理シミュレーションの場合には，タイムマップと呼ばれる構造が効率的であるといわれている[3),18)]．そこで，並列論理シミュレーションのスケジューリングキュー構造としてもタイムマップ構造が適切と考えられる．

☆ このほか，分割された部分回路全体をプロセスと考える方法もある．しかし，この方法では，プロセスに遅れてメッセージが到着した際，部分回路全体を巻き戻す必要があり，ロールバックコストがきわめて大きいと考えられる[1]ことから一般には用いられていない．

☆☆ 並列イベントシミュレーションを前提とした研究例はほとんど報告されておらず，わずかに文献4)があるのみである．

さらに，スケジューリング対象については，

- メッセージをスケジュール
- ゲートプロセスをスケジュール（メッセージは受信ゲートプロセスで管理）

の2つの選択肢がある．前者は逐次論理シミュレーションの素朴な拡張といえる．この方法では，メッセージがプロセッサに到着すると，まずスケジューリングキューに登録される．登録されたメッセージは適切なタイミングで取り出され，目的地ゲートプロセスで処理される．一方，後者は文献8)で紹介されているものである．この方法では，プロセッサに到着したメッセージは直接ゲートプロセスが受信する．受信メッセージはゲートプロセスの入力キューに保持される．スケジューラは，それぞれの入力キュー内にある未処理メッセージの最小時刻に応じて，ゲートプロセスをスケジュールする．

現在，タイムワープ機構を用いたイベントシミュレータの多くは後者を用いている[12),16)]．以後，前者をメッセージスケジューリング，後者をゲートスケジューリングと呼ぶことにする．

3.2 処理効率とデータ構造の考察

タイムワープ機構では，メッセージや履歴など実行時に動的に生成されるデータ操作が必要である．これら動的データは，一般にリスト（linked list）構造[19)]を用いて管理される．

ここで，一方向リスト（one-way list）など単純なデータ構造により動的データ管理を行うと，ロールバックやメッセージ取消に関する操作の効率が悪くなる．また，高い処理効率を得ようとすると双方向リスト（doubly-linked list）など複雑なリスト構造が必要になる．以上の点は，スケジューラを考慮した場合，特に問題となり，タイムワープ機構を用いた並列論理シミュレータのプログラミングを複雑にする一因と考えられる．

以下，最小時刻優先のスケジューリング戦略，タイムマップ構造のスケジューリングキュー，アンチメッセージ削減機構の導入，の3点を前提とし，メッセージスケジューリングおよびゲートスケジューリングそれぞれについて，処理効率およびデータ構造を例を用いて考察する．

3.2.1 メッセージスケジューリングの場合

スケジューラでは，メッセージ登録処理とメッセージ取り出し処理を行う．タイムマップ構造のスケジューラは時刻に対応したスロットからなり，メッセージは時刻に対応するスロットに登録される．

ここで，アンチメッセージ受信時のデータ操作を考察する．このとき，取り消し対象となるメッセージをすべて消去しなければならない．消去対象のうち，未処理メッセージはスケジューリングキューに存在するが，これらを取り消す処理は容易ではない．**図2**に，一方向リストを用いたデータ構造の例を示す．この場合，アンチメッセージの時刻以上のスロットをすべて探し，消去対象を見つけなければならない．

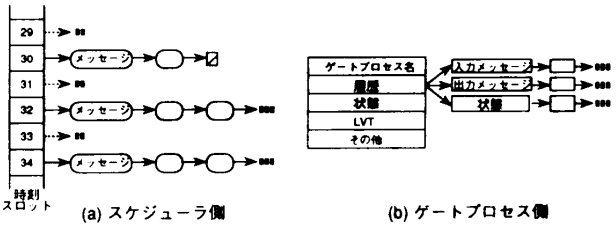


図2 メッセージスケジューリングのデータ構造例（一方向リストによる）

Fig. 2 Example of data structures for message scheduling (using one-way list).

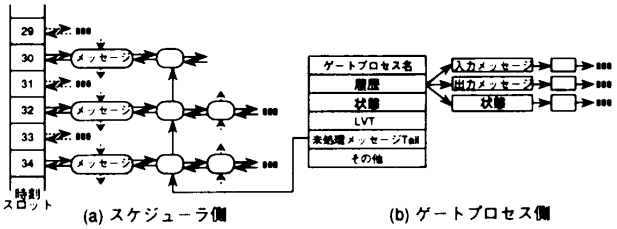


図3 メッセージスケジューリングのデータ構造例（双方向リストによる）

Fig. 3 Examples of data structures for message scheduling (using doubly-linked list).

この処理の効率化には，双方向リスト構造を用いたデータ構造が有効となる．**図3**に例を示す．ここでは，スケジューラに登録された未処理メッセージに対し，各ゲートプロセスからも直接アクセスできるようポインタを持たせることでスケジューラ内の消去対象メッセージを高速に取り消すことができる．また，同一スロット内のメッセージは双方向リストにしておくことで，リスト中の一部のみを削除する操作が効率的になる．

3.2.2 ゲートスケジューリングの場合

ゲートスケジューリングの場合，スケジューラでは，ゲートプロセス登録処理とゲートプロセス取り出し処理が行われる．ここで，アンチメッセージ受信時，およびロールバック時のデータ操作を考察する．一方向リストを用いたデータ構造の例を**図4**に示す．

アンチメッセージ受信時には，取り消し対象となるメッセージをすべて消去しなければならない．このため，未処理メッセージの入力キューを先頭から最後まで探す必要がある．

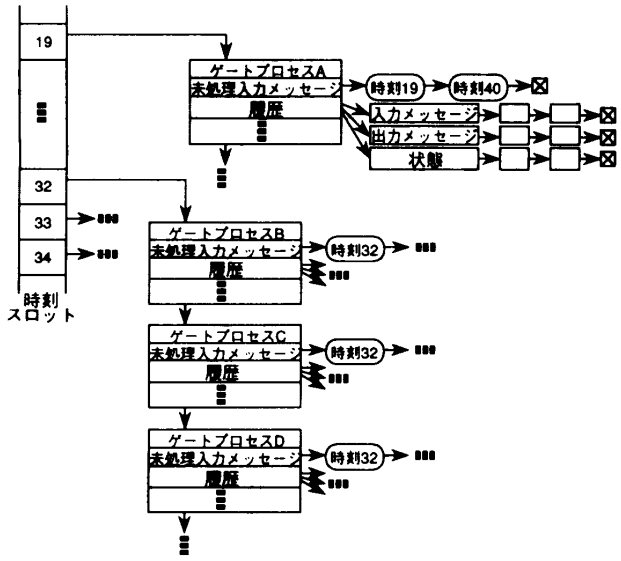


図4 ゲートスケジューリングのデータ構造例（一方向リストによる）
Fig. 4 Example of data structures for gate scheduling (using one-way list).

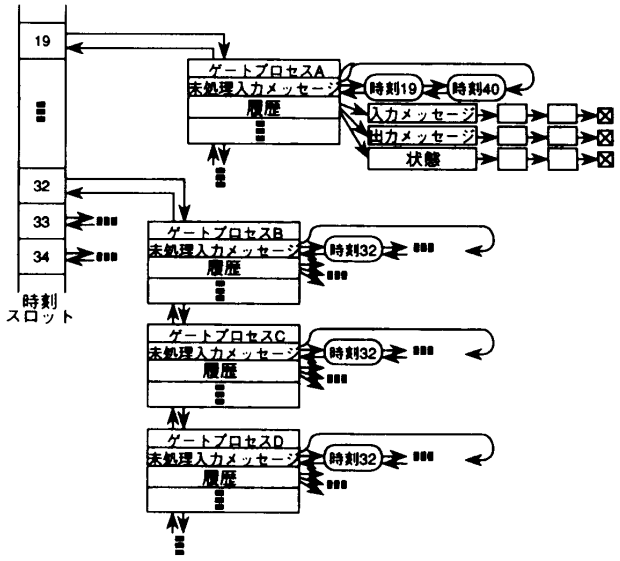


図5 ゲートスケジューリングのデータ構造例（双方向リストによる）
Fig. 5 Example of data structures for gate scheduling (using doubly-linked list).

一方，ロールバック時には，スケジューリングキュー内のゲートプロセス登録位置の変更が必要である．すなわち，現在登録中のスロットからゲートプロセスを削除し，新しい受信メッセージ時刻に対応するスロットに再登録する．図4の場合，登録ゲートプロセス削除においてスロット先頭からの逐次探索が必要になり，効率が悪い．

上記処理の効率化には，やはり双方向リスト構造を用いたデータ構造が有効である．図5に例を示す．まず，未処理メッセージ用の入力キューを双方向リストとすることにより，キューの最後尾から前方への手繰り操作が可能になる☆．これにより，時刻が十分に大きいアンチメッセージ受信時には，入力キューの先頭付近の探索が不要になるなど処理効率が向上する．一方，同時刻スロットに登録されているゲートプロセス間を双方向接続することで，登録ゲートプロセスの削除が効率的になる．

以上の考察をまとめると，メッセージスケジューリング，ゲートスケジューリングいずれの場合でも，一方向リストを用いたデータ構造ではメッセージ取消処理や再スケジューリング処理が非効率であり，その効率化のためには，双方向リストを用いたデータ構造が必要となるといえる．

C言語など，通常の手続き型言語でプログラミングを行えば，上記のデータ構造を記述することは可能である．しかし，一方向リストに比べ，双方向リストは複雑なデータ構造といえる．したがって，プログラミング容易化の観点からは，可能な限り一方向リストで効率的処理を実現できる方が望ましい．また，単一代入言語[17]でプログラム記述する場合には，双方向リストによる処理効率化は一般に困難である．

4. RCTW

本節では，上記問題の解決を目的としたタイムワープ機構の容易な実現手法であるRCTW（Rollback Counting Time Warp）を説明する．RCTWは，

1: 優先イベントなしの性質を持つ対象問題
2: 通信路のFIFO性なし
3: アンチメッセージ削減機構の導入

を前提とする．

以下，RCTWの鍵となる「ロールバック番号」および「評価済み番号」について説明した後，RCTWのアルゴリズムおよびデータ構造を説明するとともに，処理の具体例を示す．

4.1 ロールバック番号と評価済み番号

3.2.1項では，メッセージスケジューリングでの問題点を指摘した．この問題点はプロセッサへのアンチメッセージ到着時に，即座にメッセージ取消処理を行うことに起因していた．そこで，アンチメッセージ到着時ではなく，ゲートプロセスがスケジューラからイベントメッセージを受けとったときに取消判断をすることを考える．取消処理はイベントメッセージに対して行われる操作であるから，すべてのイベントメッセー

☆ もちろん，通常のメッセージ処理では，キューの前方から後方への手繰り操作が必要となる．

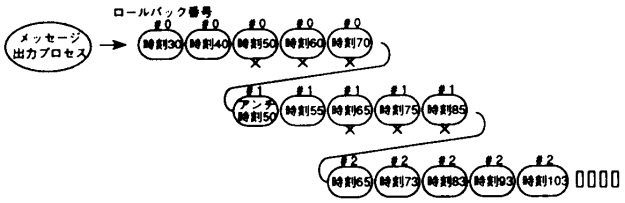


図6 ロールバック番号と出力メッセージ列

Fig. 6 Examples of rollback numbers and an output message sequence.

ジ処理時に取消対象かどうかが正しく判断できれば，すべての取消処理は正当である．

優先イベントなしの性質から，1つのゲートプロセスでは，ロールバックが発生しない限り時刻順にメッセージを送信する．そこで，ロールバック間の一連の出力メッセージ系列を1つのグループと考え，それぞれのグループに対し，発生済みロールバック回数に応じた番号（以下「ロールバック番号」と呼ぶ）をつけることを考える（**図6**）．M がロールバック番号 n のメッセージグループに属することを $M(n)$ と表すことにする．

アンチメッセージ削減機構を導入すれば，上記メッセージ列の先頭は（ロールバック番号0のグループを除き），必ずアンチメッセージである．アンチメッセージ $AM(n)$ の取消対象メッセージ $M(n')$ が満たす条件は，$n' < n$ であり，かつ，($M(n')$ の時刻) $\geq$ ($AM(n)$ の時刻) であることとなる．これは定理1から明らかである．

次に，ゲートプロセス P_A から P_B へのメッセージ通信を考える．ここで，P_A のメッセージ送信時には，ロールバック番号をすべてのメッセージに付加し，P_B でのメッセージ受信時には，受信メッセージの持つロールバック番号を記録することとする．受信側 P_B で，直前に処理した P_A からのメッセージのロールバック番号を「評価済み番号」と呼ぶことにする．評価済み番号は，すでにそのロールバック番号を持つアンチメッセージが受信されたことを意味する．したがって，受信メッセージのロールバック番号が評価済み番号より小さければ，それは取消対象であることを示す☆．なお，複数のゲートプロセスからメッセージを受信する場合には，送信元プロセスごとの評価済み番号をすべて管理しておけばよい．

以上の性質は，プロセッサ間でのメッセージ送受信の順序が保存されない環境でも成り立つことは明らかであろう．

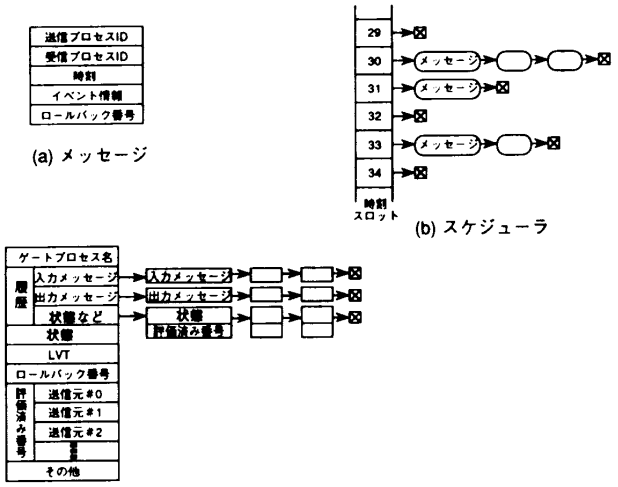


図7 RCTWにおけるデータ構造の例

Fig. 7 Examples of data structures for RCTW.

4.2 アルゴリズムとデータ構造

上記のアイデアに基づいたタイムワープ機構の実現手法がRCTWであり，以下にデータ構造とアルゴリズムを述べる．なお，説明を簡単にするため，1つのゲートプロセスは，複数の同時刻のメッセージを受信しないことを仮定する．

4.2.1 データ構造

RCTWにおけるゲートプロセスは，従来のタイムワープ機構[8]と同様のデータ（LVT，状態，入力キュー，出力キュー，状態キュー）に加え，ロールバック番号および評価済み番号という変数を持つ．すでに述べたように，ロールバック番号および評価済み番号によって，処理するメッセージが取り消し対象かどうかの判断ができる．

ロールバック時には，評価済み番号も含め，遅れメッセージが正しい順で到着したときの状況（換言すれば，履歴中最大時刻の受信メッセージ処理直後の状況）を再現する．そこで，履歴として，入力メッセージ，出力メッセージ，ゲートプロセスの状態に加えて，メッセージ処理時の評価済み番号を保存する．これらは，いずれも時刻順にソートされ，スタック構造で保持することができる．ゲートプロセスのデータ構造例およびメッセージのデータ構造例を**図7**に示す．ここでは，状態と評価済み番号をまとめて保存する例を示す．

4.2.2 アルゴリズム

RCTWでは，以下に示した流れに従って，スケジューラから取り出されたメッセージを処理する．

1: ロールバック判定

スケジューラから取り出されたメッセージ（M とす

☆ ここで，評価済み番号についても履歴保存し，P_B でのロールバック発生時に正しく回復することが必要なのは当然である．

```
0 【初期化処理】
1 (全ゲートプロセスについて以下の処理をする) {
2   全入力線 i について  E(i) := 0;
3   LVT := −1;
4   ロールバック番号 := 0;
5 }
6
7 【メッセージ評価処理】
8 繰り返し (終了条件満たすまで) {
9   if(T(受信メッセージ M) < (≤)† LVT) {          /* ロールバック判断 */
10    ロールバック番号 := ロールバック番号 + 1;
11    ロールバック処理を呼び出す;
12  }
13  if(R(M) < E(S(M)) ) {                          /* メッセージ取消判定 */
14    なにもしない;                                /* メッセージ取消 */
15  } else {
16    M, ゲートプロセス状態, E(S(M)) の履歴保存;
17    イベント処理;                                /* アンチメッセージなら何もしない */
18    出力メッセージあれば送信し, そのアンチメッセージ履歴保存;
19    E(S(M)) := R(M));                            /* 評価済み番号更新 */
20    LVT := T(M);                                 /* LVT 更新 */
21  }
22 }
23
24 【ロールバック処理】
25 繰り返し (履歴 top の入力メッセージ (M_top) 時刻 ≤ (<)† T(M) になるまで) {
26   if(S(M_top) == S(M) かつ R(M_top) < R(M)) { /* メッセージ取消判定 */
27     何もしない;                                /* メッセージ取消 */
28   } else {
29     M_top をスケジューラに再登録;
30   }
31   M_top に対するアンチメッセージあれば, Anti(D(M_top)):=アンチメッセージ;
32   E'(S(M_top)):= 履歴 top の評価済み番号;
33   M_top に対応する全履歴 (状態ほか) 消去;
34 }
35 各出力先 o について, Anti(o) にアンチメッセージあれば送信;
36 各入力元 i について, E(i) := E'(i);
```

†( ) 内は M がアンチメッセージの場合の判断基準

図 8 RCTW のアルゴリズム
Fig. 8 The RCTW algorithm.

る) は, その時刻を LVT と比較し, ロールバックが必要かどうかを判断する. M がイベントメッセージであれば M の時刻が LVT より小さい場合, また, M がアンチメッセージであれば M の時刻が LVT 以下の場合, ロールバック番号を 1 増すとともに「1': ロールバック処理」を行った後,「2: 取消判定」に進む. ロールバック処理しない場合は即座に「2: 取消判定」に進む.

1': ロールバック処理

ロールバック処理では, M がイベントメッセージの場合, M の時刻以上の履歴 (状態キュー, 入力キュー, 出力キュー) を, M がアンチメッセージの場合, M の時刻より大きい履歴を巻き戻す.

巻き戻された入力メッセージのうち, M のロールバック番号より小さいものは消去する (メッセージ取消). それ以外はスケジューラに再登録する. また, 巻き戻された履歴中に送信メッセージが保存されていれば, それらの中の最小時刻のものを記録しておく. これらは巻き戻し操作が終わった時点でアンチメッセージとして送信される (1 つの出力先に対したかだか 1 つのみ). さらに, 巻き戻されたもののうち最も過去の履歴内容に基づき, ゲートプロセスの状態および評価済み番号を回復する.

2: 取消判定

M が取り消し対象かどうかを判定する. この判定は, メッセージのロールバック番号と, M 送信元に対応する評価済み番号を比較することで行う. 後者の方が大きければ, M を取り消すアンチメッセージが送信済みであることを示しているため, M を取り消し, イベント処理は行わない. それ以外の場合はメッセージは取消対象ではないと判断し, 次のイベント処理に進む.

3: イベント処理

受信メッセージがイベントメッセージの場合には, その内容に応じた処理 (ゲートプロセス状態変化など) を行う. また, 出力メッセージがあれば送信する. さらに, 受信メッセージのロールバック番号を新しい評価済み番号として設定するとともに, 送受信メッセージやゲートプロセス状態, および評価済み番号を履歴として保存する.

なお, 1: ロールバック判定におけるロールバックが必要かどうかの判断, および 2: ロールバック処理における巻き戻しするかどうかの判断について, その判断基準がメッセージ M の種類に依存するのは以下の理由による. M がアンチメッセージの場合, これによって取り消されるイベントメッセージが履歴中に存在する可能性がある. したがって, M と同じ時刻

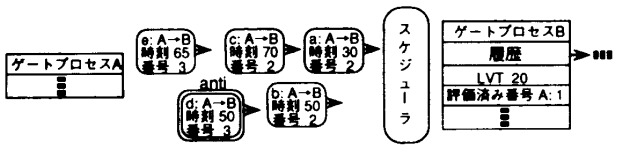


図9 RCTWにおけるメッセージ取消の例（メッセージ送信時）
Fig. 9 An example of message transmission.

のメッセージは巻き戻さなければならない．一方，M がイベントメッセージの場合，これを取り消すアンチメッセージがすでに受信されていれば，ロールバックしなくとも2: 取消判定において正しく取り消される．

図8 に，1つのゲートプロセスでのメッセージ処理に関するRCTWのアルゴリズムを示す．なおアルゴリズム表記中，以下の記号を用いている．

$S(M)$: メッセージ M の送信元ゲートプロセスの識別番号

$R(M)$: メッセージ M に付加されたロールバック番号

$T(M)$: メッセージ M の時刻

$E(S)$: 識別番号 S の送信元ゲートプロセスに対応した評価済み番号

$Anti(o)$ は，ロールバック処理において，識別番号 o の送信先ゲートプロセスに対する，最小時刻の出力メッセージを得るための作業領域である．$E'(i)$ は，ロールバック処理において，識別番号 i の送信元ゲートプロセスに関する，評価済み番号回復のための作業領域である．

4.3 メッセージ処理の例

RCTWによって正しくメッセージの取消処理が行われることを，具体例を用いて説明する．なお，以下で用いる図では，説明上不要なデータは省略している．

図9 は，ゲートプロセスAからBに，メッセージがa，b，c，d，eの順に送信された例を示している．ここでAでは，c出力後ロールバックが発生し，メッセージb，cを取り消すアンチメッセージdが送信されている．その後，イベントメッセージeを送信している．送信されたメッセージはスケジューラに登録された後，いずれゲートプロセスBで評価される．いま，これらのメッセージ処理前には，BはLVT 20，Aに関する評価済み番号2という状態にあるとし，以下の例について考える．

例 Bがa，b，d，c，eの順にメッセージ処理する場合（**図10**）．

このような状況は，通信路のFIFO性の有無にかかわらず起こりうる．たとえば，

1: a，bをスケジューラが受信

2: a，bを目的地ゲートプロセスが処理

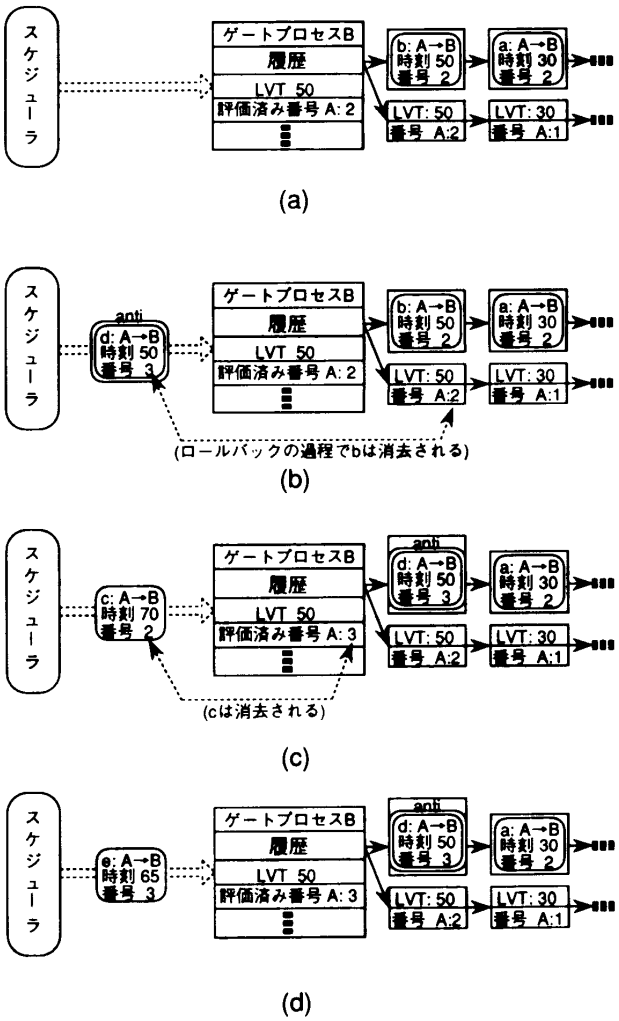


図10 RCTWにおけるメッセージ取消の例
Fig. 10 An example of message cancellation.

3: c，dをスケジューラが受信

4: d，cを目的地ゲートプロセスが処理

5: eをスケジューラが受信

6: eを目的地ゲートプロセスが処理

という順に処理が進んだ場合である．

最初のa，bについては，ロールバックやメッセージ取消されることなくイベント処理される．これらの処理直後には，LVTは50，Aに対応する評価済み番号は2となっている（図10(a)）．次にdが処理される．このとき，dの時刻がLVT以下のため，ロールバックが発生する（図10(b)）．ロールバックにより履歴スタック中のメッセージbは巻き戻される．ここで，ロールバック番号はdよりbの方が小さいため，bは消去される（図8: 27行め）．ロールバック処理の後，dが評価され（アンチメッセージなのでイベント処理は行われない），Aの評価済み番号が3に更新される．

続いてメッセージcの処理が行われる（図10(c)）．

このとき，cのロールバック番号は送信元Aに対応する評価済み番号より小さいため，cは取り消される．最後にメッセージeが処理される．このロールバック番号は送信元Aに対応する評価済み番号以上であるため，eは取り消されることなくイベント処理される(図10(d))．

以上から，RCTWによって正しくメッセージ取消処理が行われることが直観的に理解できよう．なお，RCTWの厳密な正当性については，付録に証明を記す．

5. 議　　論

5.1 データ構造

3章で示した例では，メッセージ取消処理，あるいはゲートプロセスの再スケジュールを効率的に行うためには，双方向リストに基づくデータ構造を用いる必要があった．これに対しRCTWでは，図7で示した一方向リスト（スタック構造）で効率的な操作が実現される．

この性質は，オペレーティングシステムや言語で十分なレベル数を持つ優先度機構がサポートされている場合[14]，それをメッセージスケジューラとして利用できることを示している．一般に，OSや言語で提供されている優先度機構では，いったんスケジュールされたイベントは取り消すことができないが，この性質はRCTWでは問題とならない．これはRCTWの大きな利点といえ，並列シミュレータの設計・開発コストを，より低くすると期待される．

5.2 メモリ効率

RCTWでは，ロールバック番号や評価済み番号などの新しい変数を導入することから，3章で示した例に比べメモリ使用量が増加する．しかし，もともと必要なデータ（時刻，状態，イベント情報など）の量に比べ，新たに必要なデータ量は少ないため，メモリ使用量増加の割合は小さいと考えられる．

5.3 処理効率

(1)　新しい変数操作に関して

RCTWでは，ロールバック番号や評価済み番号に関する新たな操作が処理効率を低下させると懸念される．しかし，一方で，3章で述べたタイムワープ機構実現手法では，複雑なリスト操作が必要となる．

この違いによる処理コストの差はプログラミング技術や使用する計算機の性質に依存するため，結論づけることは難しい．ただし，これらの処理は，メッセージの生成やイベント処理などシミュレーションの本質的な処理に比べると大きくなく，実際には，上記処理コストの差はほとんどないと考えられる．

(2)　スケジューリングポリシの違いに関して

3章で述べた手法とRCTWとでは，メッセージ取消のタイミングが異なる．すなわち，3章の手法ではアンチメッセージ受信時に，即座に取消対象メッセージを取り消し，必要があればさらに次の出力先へアンチメッセージを送信する．これに対し，RCTWでは，アンチメッセージ受信時には，いったんスケジューラに登録し，それを処理するときに初めて取消処理が行われる．したがって，RCTWの方がアンチメッセージの伝播が遅れる傾向にある．

この違いはシミュレータの処理速度に影響することが考えられる．しかし，必ずしもRCTWが不利とはいえない．RCTWでは，アンチメッセージ AM 受信時に，AM より先にスケジュールされるべきイベントメッセージ M があれば，M を先に処理するべきと考えている．

これに対し，3章の手法では，アンチメッセージ AM を受信すれば，すべてに優先して AM に関する処理を行う．しかし，AM より時刻の小さいイベントメッセージ M があるのにもかかわらず，M の処理を遅らせれば，結局，M の出力送信先でロールバックを発生させる可能性が高くなると思われる．

以上の違いは，スケジューリングポリシの違いといえるが，上記考察から，筆者らはRCTWのスケジューリングポリシの方が適切であると考える．

6. おわりに

スケジューラを考慮したタイムワープ機構の効率的な実現手法として，RCTW（Rollback-Counting Time Warp）を提案した．RCTWは，各プロセスにて発生したロールバック数をメッセージに付加し，受信時にこれを比較するという操作を新たに導入している．この単純な操作の追加により，以下の2点が可能になった．1: アンチメッセージ受信時即座の取消処理を不要とし，かわりにイベントメッセージ処理時に取消対象かどうかの判断を行うことにより，データ構造が単純化された．2: 従来は，プロセス間のメッセージ送受信においてメッセージ順序が保存される環境でのみ適用可能であったアンチメッセージ削減機構を，順序保存性のない環境においても適用可能となった．これらの点は，シミュレータの設計・開発コスト低減に貢献すると考えられる．さらに，汎用的なロールバックオーバヘッド削減機構も実現しているといえる．

また，本稿では，RCTWの応用を論理シミュレーションとして説明したが，対象問題が優先イベントなしの性質を持ち，スケジューリングキューとしてタイ

ムマップ構造が有効であれば，RCTW はそのまま適用可能である．

なお，RCTW の課題としては，ロールバック番号オーバフローの問題がある．この問題は，番号オーバフロー検出時に，1: 全プロセスの時刻を GVT まで巻き戻す，2: 全プロセスのロールバック番号，および評価済み番号を初期化する，3: スケジューラを初期化する，という 3 つのステップを経た後，GVT からシミュレーションをやり直すことで解決可能である．しかし，ロールバック番号はプロセスにロールバックが発生するとき 1 ずつ増加するのみであることを考慮すれば，上記の 3 ステップの処理を組み込まなくても実用上の問題はないと考えられる．

謝辞 本稿の作成に関し，多くの貴重な助言をいただいた京都大学中島浩助教授に深謝します．また，本研究を行うにあたり，有益な助言をいただいた元 ICOT PIC-WG 委員諸氏に感謝します．

参 考 文 献

1) Briner, J., et al.: Parallel Mixed-level Simulation Using Virtual Time, *CAD Accelerators* (Ambler, A., et al.(eds.)), North-Holland, pp.273–285 (1991).
2) Burdorf, C. and Marti, J.: Non-Preemptive Time Warp Scheduling Algorithms, *ACM Operating System Review*, Vol.24, No.2, pp.7–18 (1990).
3) d'Abreu, M.: Gate-Level Simulation, *IEEE Design & Test of Computers*, Vol.2, No.6, pp.63–71 (1985).
4) Das, S. and Fujimoto, R.: A Performance Study of the Cancelback Protocol for Time Warp, *7th Workshop on Parallel and Distributed Simulation*, pp.135–142 (1993).
5) Fujimoto, R.: Parallel Discrete Event Simulation, *Comm. ACM*, Vol.33, No.10, pp.30–53 (1990).
6) 福井眞吾：バーチャルタイムアルゴリズムの改良，情報処理学会論文誌，Vol.30, No.12, pp.1547–1554 (1989).
7) Gafni, A.: Rollback Mechanisms for Optimistic Distributed Simulation Systems, *Distributed Simulation '88*, pp.61–67 (1988).
8) Jefferson, D.: Virtual Time, *ACM Trans. Prog. Lang. Syst.*, Vol.7, No.3, pp.404–425 (1985).
9) Jefferson, D., et al.: Implementation of Time Warp on the Caltech Hypercube, *Distributed Simulation '85*, pp.70–74 (1985).
10) Matsumoto, Y. and Taki, K.: Adaptive Time-Ceiling for Efficient Parallel Discrete Event Simulation, *Western Multiconf. on Computer Simulation—Object-oriented Simulation*, pp.101–106 (1993).
11) 松本幸則，瀧 和男：バーチャルタイムによる並列論理シミュレーション，情報処理学会論文誌，Vol.33, No.3, pp.387–395 (1992).
12) McBrayer, T.: Personal Communication, (1995).
13) Misra, J.: Distributed Discrete-Event Simulation, *ACM Computing Surveys*, Vol.18, No.1, pp.39–64 (1986).
14) 小倉 毅，瀧 和男：並列オブジェクト指向言語 mosaic とマルチワークステーション上の実装，*JSPP '94*, pp.97–104 (1994).
15) Sokol, L., et al.: MTW: A Strategy for Scheduling Discrete Simulation Events for Concurrent Execution, *Distributed Simulation '88*, pp.34–42. (1988).
16) Steinman, J.: Personal Communication, (1995).
17) Ueda, K. and Chikayama, T.: Design of the Kernel Language for the Parallel Inference Machine, *Comput. J.*, Vol.3, No.6, pp. 494–500 (1990).
18) Ulrich, E.: Event Manipulation for Discrete Simulations Requiring Large Numbers of Events, *Comm. ACM*, Vol.21, No.9, pp.777–785 (1978).
19) たとえば浦 昭二：データ構造，共立出版 (1974).

付録 RCTW におけるメッセージ取消処理の正当性

対象問題において，優先イベントなし，および 1 つのゲートプロセスに対し同時刻のイベントが発生しないという 2 つの性質があること，さらに通信路の FIFO 性がない環境を前提に，RCTW の正当性を証明する．以下，イベントに対応したメッセージをイベントメッセージと呼ぶ．単にメッセージと呼ぶ場合は，イベントメッセージおよびアンチメッセージともに含めて考える場合である．また証明過程において参照されている行数は，すべて図 8 における行数である．

イベントメッセージ EM が取り消される場合に満たすべき条件（取消条件と呼ぶ）は，定理 1 を書き換え，以下のようになる．

取消条件 イベントメッセージ EM に対し，$S(EM) = S(AM)$, $R(EM) < R(AM)$, $T(EM) \geq T(AM)$ なるアンチメッセージ AM が存在する．

RCTW におけるメッセージ取消処理の正当性を証明するには，1: イベントメッセージ EM が取消条件を満たす場合，かならず EM は送信先ゲートプロセ

スにおいて取り消されること，および，2: イベントメッセージ EM が取り消される場合，EM は必ず取消条件を満たすこと，を示せばよい．まず，3つの補題を証明し，続いて，2: に対応した定理5，および1: に対応した定理6を証明する．

補題2 2つのメッセージ M_a, M_b において，$S(M_a) = S(M_b)$, $T(M_a) \geq T(M_b)$, $R(M_a) < R(M_b)$ なら，$S(M_a) = S(AM)$, $T(M_a) \geq T(AM)$, $R(M_a) < R(AM)$ となるアンチメッセージ AM が存在する．

(証明) $R(M_a) < R(M_b)$ より，M_b が M_a のアンチメッセージであるか，M_b, M_a ともに互いのアンチメッセージでないか，のいずれかである．前者の場合，$AM = M_b$ となり補題2が成立．後者の場合，M_b, M_a は異なるイベントメッセージまたはそのアンチメッセージとなる．そこで，送信側においてイベントメッセージ m_a, m_b を受信し，その出力イベントメッセージとして，あるいはそれらのアンチメッセージとして M_a, M_b を送信したとする．優先イベントなしの性質 ("$T(em_a) < T(em_b)$ なら $T(EM_a) < T(EM_b)$") の対遇から，$T(M_b) < T(M_a)$ ならば $T(m_b) < T(m_a)$ (同時刻イベントなしの前提から $T(M_b) \neq T(M_a)$ かつ $T(m_b) \neq T(m_a)$)．

$R(M_a) < R(M_b)$ から，送信側では M_a, M_b の順に送信している．また，これは M_b 送信前に，m_a をいったん受信したことを意味する．$T(m_b) \leq T(m_a)$ であるから，M_a 送信後，M_b 送信時までに，m_a は少なくとも一度はロールバック処理で巻き戻されている．したがって，このロールバックにおいて，31行めの操作により $Anti(\ )$ に最終的に書き込まれているメッセージ AM は $T(AM) \leq T(M_a)$ となる．このロールバックは M_a 送信後に発生しているため，$R(AM) > R(M_a)$ となることは明らか．(証明終)

補題3 メッセージ取消判断を行う時点 (13行め) で，ゲートプロセス P_{src} から送信されたメッセージで履歴内に保存されているもののうち，最大ロールバック番号を持つものを $M_{maxR:hst}$ とすると，$E(P_{src}) = R(M_{maxR:hst})$ である．ただし，履歴中に P_{src} から送信されたメッセージがない場合，$R(M_{maxR:hst}) = 0$ とする．

(証明) 帰納的に証明する．

初期状態では $E(P_{src}) = 0$，履歴は空であるから，補題3が成立．

メッセージ評価処理の中で $E(P_{src})$ や履歴領域が変化するのは，ロールバック処理呼び出し (11行め) の前後，および，履歴保存や評価済み番号更新の処理 (16〜19行め) の前後のみである．そこで，これらのカ所の実行前に補題3が成り立てば，実行後も成り立つことを示す．

履歴保存および評価済み番号更新の処理の前後 — 処理メッセージ M が P_{src} 以外から送信されたものであれば，$E(P_{src})$, $M_{maxR:hst}$ とも変化しないため補題3は成立．処理メッセージ M が P_{src} から送信されたものであれば，13行めの判断時点で $E(P_{src}) = R(M_{maxR:hst}) \geq R(M)$ である (さもなければ16行め以降の処理が行われない)．したがって，19行めの時点では M が履歴中最大のロールバック番号となることは明らか．評価済み番号更新 (19行め) により $E(P_{src}) = R(M)$ なので，補題3は成立．

ロールバック処理の前後 — 最初のロールバックが発生する直前までは，16〜19行めによる変化のみであるため補題3が成立．ここで，n 回めのロールバック処理の前後を考える．ロールバック後の履歴の状態および評価済み番号は，最後に巻き戻されたメッセージを受信する直前のものに等しいように回復される．したがって n 回めのロールバック発生直前まで補題3が成立していれば，n 回めのロールバック直後も成立．

以上から，つねに補題3が成り立つ．(証明終)

補題4 メッセージ M が取り消される場合，M は以下の拡張取消条件を満たす．

拡張取消条件：メッセージ M に対し，$R(M) < R(AM), T(M) \geq T(AM), S(M) = S(AM)$ なる AM が存在する．

(証明) メッセージの取消は，14行め，あるいは27行めのどちらかでのみ行われる．いずれの場合でも補題4が成立することを示す．

27行めの取消処理 — ロールバックを起こした受信メッセージを M_r とする．メッセージ M が保存されている履歴が巻き戻されることから $T(M_r) \leq T(M)$. 27行めによって M を消去することから $S(M_r) = S(M)$ かつ $R(M_r) > R(M)$. ゆえに補題2より，$S(M) = S(AM), T(M) \geq T(AM), R(M) < R(AM)$ となるアンチメッセージ AM が存在．M は明らかに拡張取消条件を満たす．

14行めの取消処理 — 履歴中に M と同じ送信元からのメッセージがなければ，補題3から $E(P_{src}) = 0$ (初期値) である．このとき M は取り消されないため，前提に反する．ゆえに，履歴中に必ず $S(M_{hst}) = S(M)$ となる入力メッセージ M_{hst} が存在．

履歴中にある，M と同じ送信元からのメッセージのうち，最大のロールバック番号を持つものを $M_{maxR:hst}$ とすると，補題3から $E(P_{src}) =$

$R(M_{maxR:hst})$. いま，$R(M) \geq R(M_{maxR:hst})$ と仮定すると，M 評価の直前は $E(P_{src}) = R(M_{maxR:hst}) \leq R(M)$ となり，M が取り消されるという前提と矛盾する．ゆえに $R(M) < R(M_{maxR:hst})$. $M_{maxR:hst}$ は履歴にあるため，$T(M) \geq T(M_{maxR:hst})$. また，$S(M) = S(M_{maxR:hst})$ であるから，補題 2 から $T(M) \geq T(AM)$，$R(M) < R(AM)$ となるアンチメッセージ AM が存在．ゆえに M は拡張取消条件を満たす．(証明終)

補題 4 において M をイベントメッセージ EM とおけば次の定理となる．

定理 5 イベントメッセージ EM が取り消される場合，EM は取消条件を満たす．

また，補題 4 の M をアンチメッセージ AM に置き換えれば，AM が取り消される場合，AM は拡張取消条件を満たすことがいえる．この対遇をとれば次の系となる．

系 5.1 アンチメッセージ AM が拡張取消条件を満たさなければ，AM は取り消されない．

定理 6 イベントメッセージ EM が取消条件を満たす場合，EM は取り消される．

(証明) いま，EM の送信元ゲートプロセス P_{src} から，EM 以降に送られたアンチメッセージの中で最小時刻を持つものを AM_{minT} とする．AM_{minT} は，その定義から，拡張取消条件を満たさないことは明らか．ゆえに，系 5.1 から AM_{minT} は取り消されない．また，EM が取消条件を満たすこと，および AM_{minT} の定義から，$T(EM) \geq T(AM_{minT})$, $R(EM) < R(AM_{minT})$.

ここで，AM_{minT} と EM の処理順序は，場合 1：EM の後 AM_{minT}，場合 2：AM_{minT} の後 EM，のどちらかである．

場合 1 — EM の処理時に EM が消去されれば定理 6 が成立．消去されなければ，AM_{minT} の処理開始時に，履歴中に EM が存在．したがって，ゲートプロセスの LVT は $T(EM)$ 以上であり，ロールバックが発生する．ロールバック処理の過程 (27 行め) で EM は取り消されるため，定理 6 が成立．

場合 2 — AM_{minT} は取り消されないため，EM の処理開始時には履歴中に AM_{minT} が存在．EM 処理時には，AM_{minT} の履歴は巻き戻されず，履歴中に存在する．補題 3 から，$E(P_{src}) \geq R(AM_{minT})$ である．AM_{minT} の定義から $R(EM) < R(AM_{minT})$ であり，EM は取り消される．ゆえに場合 2 でも定理 6 が成立．(証明終)

定理 5 と定理 6 から，RCTW におけるメッセージ取消処理の正当性が証明された．

(平成 7 年 8 月 28 日受付)
(平成 8 年 2 月 7 日採録)

松本 幸則

昭和 37 年生．昭和 60 年京都大学工学部電子工学科卒業．同年三洋電機 (株) 入社．筑波研究所にて FA 用画像処理検査装置の開発に従事．平成元年 (財) 新世代コンピュータ技術開発機構に出向．並列応用ソフトウェアの研究開発に従事．現在，三洋電機 (株) 東京情報通信研究所に所属．工学博士．電子情報通信学会会員．

瀧 和男 (正会員)

昭和 27 年生．昭和 51 年神戸大学工学部電子工学科卒業．昭和 54 年同大学院修士課程システム工学修了．工学博士．同年 (株) 日立製作所入社．昭和 57 年 (財) 新世代コンピュータ技術開発機構に出向．逐次型および並列型推論マシンと並列応用プログラムの研究開発に従事．平成 2 年同機構第 1 研究室室長．平成 4 年 9 月神戸大学工学部情報知能工学科助教授．平成 7 年 4 月同学科教授．並列マシンのアーキテクチャ，並列プログラミング，LSI-CAD 等に興味を持つ．電子情報通信学会，IEEE，ソフトウェア科学会，ACM 各会員．