# LOTOSプロセスの並列処理環境における同期処理機構

4P-7

野村真吾　　長谷川亨　　瀧塚孝志
国際電信電話株式会社　上福岡研究所

## 1. はじめに

筆者らは、ISOで規定された仕様記述言語であるLOTOS[1]の実行環境を実現するために、LOTOSの仕様に現れる並列プロセスを実行するスケジューラを作成している。LOTOS仕様中のプロセスは、トランスレータによりC言語の関数に変換され、スケジューラ上でライトウェイト・プロセスとして実行される[2]。LOTOSのプロセス間通信は、複数のプロセスが同時に同期できる点および複雑な同期要求ができる点に特徴がある。従って、このような同期通信を実現する機構がスケジューラ実現のキーとなる。そこで本稿では、LOTOSプロセスの同期機構を実現するスケジューラの機能およびメカニズムを提案する。

## 2. LOTOSの同期通信機構

LOTOSはソートを持つ並列論理型言語である。LOTOSによる仕様は複数のプロセスから構成され、プロセスの動作は外部プロセスとの入出力応答であるイベントの列挙により記述される。イベントはソート付きの値または変数であり、ゲート(プロセス間のインタラクション点)を共有するプロセス間でユニファイ可能な時に同期して実行される。この同期通信は送信/受信プロセスを特定した1対1の同期通信とは異なり、次のような特徴を持つ。

① プロセスは2種類の並列結合オペレータを用いて、"B1||B2"または"B1|||B2"と記述することにより、複数の並列に動作する子プロセス(B1,B2)を生成できる。子プロセスは、親プロセスのゲートを共有し、親プロセスがこのゲートを介して同期する外部プロセスと同期する必要がある。

② 完全同期実行"B1||B2"を用いて生成した子プロセス(B1,B2)は、外部プロセスとだけでなく子プロセス同志も同期する必要がある。独立実行"B1|||B2"を用いて生成した子プロセスはどちらかのプロセスが外部プロセスと同期すればよい。

③ ガードを伴う構造イベントにより、複雑な同期要求が記述できる。"[x>0]->g!x?y:int[y>5]"の記述では、値xが正数である時ゲートgに対してイベントが発行され、変数yが5より大きな整数とユニファイ可能な時に限りイベントが実行される。

LOTOSの実行過程において、イベントが実行されるために同期すべきプロセスの数および相手は、動的な子プロセスの生成により、刻々と変化していく。従って、効率良い同期通信機構を実現するためには、(1)同期すべきプロセスの数および相手の管理、(2)構造イベントのユニフィケーションの2点の実現がキーとなる。

## 3. スケジューラの同期処理部

並列プロセスを管理するスケジューラとLOTOSプロセスの関係、およびそのアルゴリズムを図1に示す。

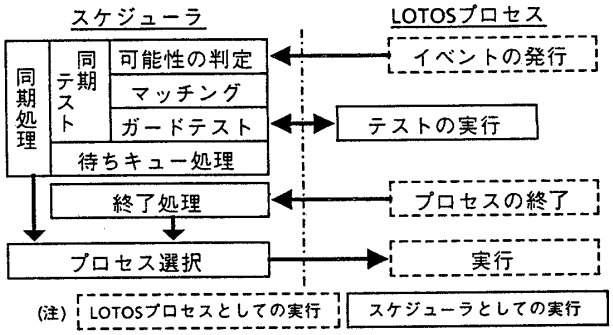


図1　スケジューラのアルゴリズム

プロセスがイベントの発行またはプロセスの終了を実行すると、スケジューラが起動される。スケジューラは、同期処理または終了処理を行い、実行可能なプロセスを選択し、選択されたプロセスに制御を渡す。プロセスは同期待ちキューと実行待ちキューを用いて管理され、プロセスはイベントを発行すると同期待ちキューにイベントをつないで、同期待ちの状態になる。

### 3.1. 同期待ちキューの構成

イベントでは、同期する相手(同期対象)を指定しないため、同期対象を探索する必要がある。同期すべきプロセスの関係を保持するために、同期待ちキューは、宣言されたゲート毎に図2に示すようなプロセスの親子関係を表現する木構造を持つ。

並列結合オペレータを用いて新しいプロセスを生成すると見かけのゲートが生成され、生成された

Implementation of Synchronized Communication Mechanism in LOTOS Process Scheduler
Shingo NOMURA, Toru HASEGAWA, Takashi TAKIZUKA
KDD Kamifukuoka R & D Laboratories

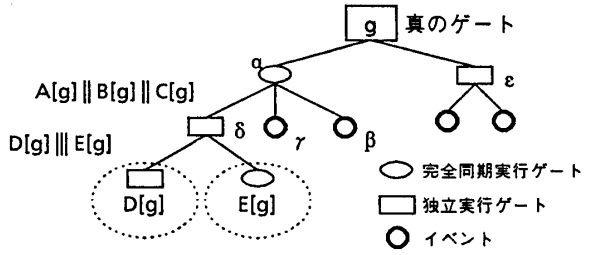


図2 同期待ちキュー

プロセスの発行するイベントは、対応する見かけのゲートにつながれる。見かけのゲートでは完全同期実行と独立実行を区別して同期関係を表現する。図2では3つのプロセスA,B,Cが完全同期により生成され、さらにプロセスAがプロセスD,Eを独立実行により生成したことを示している。

図中のイベントβにより同期が成立するためには、完全同期実行ゲートαにつながるイベントβ,γ、独立実行ゲートδを介したイベント、およびゲートgを介したゲートεにつながるイベントが全て発行されており、さらに全てのイベントがユニファイ可能である必要がある。

### 3.2. 同期テスト

同期テストでは同期の成立を確認するが、処理の効率を上げるために、同期可能性の判定、マッチングテスト、ガードテストの3つに分けて行う。

(1) 同期可能性の判定

同期待ちキューにおいて、完全同期実行の見かけのゲートを生成した時点では親のゲートとつながず、そのゲートにつながるイベントが全て発行されてから親ゲートにつなぐことにする。これにより、あるイベントを発行した時、発行されたイベントから真のゲートgまでたどれる場合のみ、同期が成立する可能性があると判定できる。

(2) マッチングテスト

マッチングテストでは、LOTOSの同期条件をテストし、同期が成立する場合には対応する後処理を実行する。この一連の動作は、Prolog等の言語に見られるユニフィケーションと酷似している。

スケジューラは、3.1で述べたように同期待ちキューをたどり同期対象を探索しながら、発行されたイベントと同期対象として選択されたイベントが同期条件を満たすかどうかをテストする。本実行環境では、データのソートを管理する目的とメモリ管理の目的から、デスクリプタを用いてデータの値を保持している。このためデータのソートの一致を確認した後で、デスクリプタをたどって値のマッチングを確認する。

(3) ガードテスト

LOTOSでは、ガードを用いてイベントに制約を付けることができる。本実行環境はC言語により実装しているため、スケジューラにガード情報を渡すことが困難であった。そのためプロセスの一部にガードの条件を展開しておき、ガードテスト時にコルーチンとして一時的にプロセスに制御を戻して、テスト結果をスケジューラに返している。

プロセスは、ガード付きのイベントであることをイベントの情報に付け加えて発行する。スケジューラは、マッチングテスト中に区別しておいたガード付きプロセスに、順に制御を渡してガードテストを行う。ガードテストは、イベントによるマッチングが終了し、値が決定している必要があるため、マッチングテストの終了後に実行する。

マッチングの実行中にテストが失敗した時、またはガードテストにおいて同期が成立しなかった時、バックトラックを行い、別の同期対象を探索して再度ユニフィケーションを行う。

### 3.3. 待ちキュー処理

イベントによる同期が成立すると、同期待ちキューにつながれているイベント情報を持つプロセスを、実行待ちキューにつなぐ。この時同時に、次の2つの処理を行う。

選択実行(B1 [] B2)により、複数個の動作を並列して選択することができる。個々の動作をプロセスとして実現することもできるが、本処理系では処理効率を考慮し、その動作がイベントである場合には、一つのプロセスから複数個のイベントが発行できるようにして実現した。このため同期待ちキューにつながれている残りのイベント情報を、同期待ちキューから取り除く。

また、LOTOSの中断実行(B1 [>B2)では、B2が実行可能となるとB1の実行を中断する。これはB2の発行したイベントによる同期が成立することにより起こるため、このプロセスの実行中断管理を待ちキュー処理で実行する。

## 4. おわりに

LOTOSの同期通信の実現では、複数のプロセス間で同期が成立すること、同期する相手が動的に変化することが問題であり、同期対象の管理が重要である。これを実現するために同期の関係を表現する構造を同期待ちキューに持たせることで実現した。

現在、本稿で述べた同期通信を実現する並列プロセス処理環境を作成中であり、また本処理環境上で動作するプログラムを生成するトランスレータについて検討中である。最後に日頃御指導頂くKDD上福岡研究所小野所長、湯口次長、通信ソフトウェア研究室小西室長、若原主任研究員、コンピュータ通信研究室加藤主査に感謝する。

参考文献

[1]: ISO/DIS 8807,Sep. 1987.
[2]: 野村,長谷川,瀧塚,"LOTOSの同期通信機構を持つ並列処理環境の構築法",情処全大,5E-2, Sep. 1988.