LOTOS実行系を用いたプログラム変換実験

6L−9

野村 真吾　瀧塚 孝志　長谷川 亨
国際電信電話株式会社　研究所

1. はじめに

LOTOS[1]は、通信プロトコルの仕様を記述するために、ISOで標準化された仕様記述言語の一つである。筆者らは、LOTOSで記述された仕様を、C言語のプログラムに変換して実行するLOTOS実行系を作成している[2]。本実行系では、変換されたプログラムの実行速度は、実行中に存在するプロセス数に依存する。そこで、プロセス数を減らして効率のよい動作を実現するため、自プロセスの生成による再帰的動作をループ構造に書き換えたり、プロセス仕様を親プロセスの仕様中に展開する等の最適化方式の検討を行っている[3]。今回、通信システムの基本的な動作を規定した小規模な仕様を、最適化を用いないでプログラムに変換し、生成されるプロセス数と実行速度を測定した。その結果、実行中に生成されるプロセスの87%が再帰的動作により生成されており、再帰的動作をループ化する最適化処理の有効性が確認できた。

2. LOTOS実行系

2.1 LOTOS実行系

本実行系は、トランスレータとスケジューラにより構成される(図1)。トランスレータは、LOTOSで記述されたプロトコル仕様を、スケジューラの機能を使用するC言語のプログラムに変換する。スケジューラは、プロセスの実行やイベントによる同期と値の送受等、LOTOSの言語仕様で規定される基本的な動作を提供する。

本実行系では、仕様に定義されたプロセスをスケジューラが管理するプロセスとして動作させるため、OSのプロセスとして動作させる場合に比較して多くのプロセス(約1400個)が動作可能である[2]。

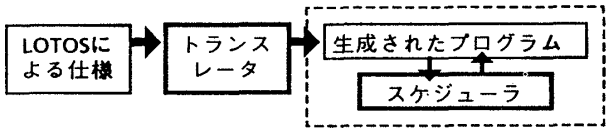

図1　LOTOS実行系の構成

2.2 LOTOS書換系

LOTOSでは、繰り返し処理を自プロセスの生成(自己再帰)または親プロセスの生成(親への再帰)として記述する。親プロセスは、子プロセスの終了を待つため、再帰的動作を含む仕様では、実行にともないプロセス数が増大していく。

LOTOS書換系は、入力仕様に対して、再帰的動作のループ化やプロセス仕様の展開を行い、生成されるプロセス数が少ない最適化仕様に書き換える。

3. プログラム変換実験

3.1 アブラカダブラプロトコル仕様

今回の変換実験では、通信システムの基本的な動作を規定したアブラカダブラプロトコルの仕様[4]を使用した。このプロトコルは、呼設定において発呼側と着呼側の2種類の動作を持つ。発呼側では、接続要求を受取り信号CRを着呼側に送信する。着呼側は、これを受信して接続指示を通知する。続いて着呼側が接続応答を受取り、信号CCを発呼側に送信する。発呼側は、これを受取り接続完了を通知する。データ転送では、受信側から送信側に受信確認の信号が転送される。

3.2 仕様変換結果

アブラカダブラプロトコルの仕様を変換した結果を表1に示す。トランスレータは、動作記述のみを変換の対象とするため、入力仕様の行数は、全仕様からデータ定義部分(134行)を除いた値である。

入力仕様	変換結果	追加記述	
		データ	インタフェース
387行	1,357行	562行	115行

表1　プログラム変換結果

変換されたプログラムを動作させるために、データ型の定義とデータに対するオペレーション、および外部プログラムとのインタフェースを追加記述した。

3.3 プログラム評価結果

上位レイヤ、下位レイヤの働きをするプログラムを作成して、生成されるプロセス数、実行時間を測定した。上位レイヤは、呼設定を行った後、ファイルを固定長のデータに分割し、150回に分けて転送する。測定は、上位レイヤ、下位レイヤを除いたアブラカダブラプロトコルのデータ送信側のプログラムに対して行った。実行時間は、同一条件のもとで、10回の測定を行い平均値を求めている。

Preliminary Evaluation of LOTOS Translation Method
Shingo NOMURA, Takashi TAKIZUKA, Toru HASEGAWA
KDD R & D Laboratories

プログラムの転送実験は、VAX Station-3100 (VAX780を1MIPSとして2.8MIPS)上で行った。上位レイヤ、下位レイヤのプログラムとは、VAX VMSのメールボックスをインタフェースとして使用している(図2)。

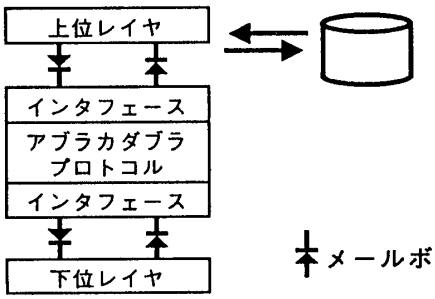

図2　実行速度の測定

(1)プロセス数

呼設定では、16個のプロセスが生成されて、4回の同期が行われる。1回のデータの転送につき8個のプロセスが生成されて、3回の同期が行われる。

呼設定からデータ転送終了までの間で、合計1,224個のプロセスが生成され、454回の同期が行われた。この間に、再帰的動作により生成されたプロセス数と、disabling等により消滅したプロセス数を表2に示す。

		プロセス数		割合(%)	
再帰による生成数	自己再帰	1,059	758	87	62
	親への再帰		301		25
消滅したプロセス数		151		12	
その他		14		1	
総数		1,224		100	

表2　生成されたプロセスの内訳

(2)実行時間の測定

接続要求を出してから、接続完了の通知を受け取るまでの時間は、約30msであった。

1回に転送するデータの長さを変えて、実行時間を測定した結果を表3に示す。アブラカダブラプロトコルではデータを操作することが少なく、また生成されたプログラムにおいてもデータのコピー等が行われないため、実行時間はデータの転送回数とメールボックス(MBX)への転送時間に依存する。

データ長 (byte/回)	総データ長 (kbyte)	実行時間(ms) (内MBX転送時間)	平均 (ms/回)
128	19.2	6,383 (168)	41.4
1,024	153.6	6,443 (244)	41.3
2,048	307.2	6,667 (365)	42.0

表3　プログラム実行速度

4. 考察

(1)プロセス数

アブラカダブラプロトコルでは、データ転送の繰り返しが再帰により記述されている。このためデータ転送中は、プロセス生成が繰り返されて、ワークステーションの標準的な環境設定のもとでは、約150回のデータ転送で計算機資源を使い果たしてしまう。この間に、disabling等で消滅したプロセスは、生成されたプロセス数の1割程度であった。

データ転送中は、3個のプロセスが自己再帰を行い、1組のプロセスが親への再帰を行っている。これらのプロセスの延べ生成数は、生成された全プロセス数の87%になる。このことから、再帰的動作をループ化して1つのプロセスで実現する方法は、実行中に存在するプロセス数を減らすのに有効であり、計算機資源を使い果たすことなく大量のデータを転送することができる。また、プロセス数が減ることにより、本実行系で作成するプログラムがさらに効率よく動作することが期待できる。

(2)プログラム規模

アブラカダブラプロトコル仕様の変換では、プログラム全体の67%が自動生成された。データ型の定義、およびデータに対するオペレーションの実現、さらに外部プログラムとのインタフェース部分を追加記述する必要があった。

データに対しては、LOTOSでのライブラリが整備されて、それを実現するプログラムライブラリを準備することにより、新たなプログラム記述の量を減らしていくことができる。またOSIのアプリケーションプロトコル等では、データの定義にASN.1等を用いており、他の開発ツールと組み合せることにより追加記述量を減らすことができる。

インタフェースに関する追加記述は、メールボックスを使用する場合、大幅な変更を必要としないで、今回の枠組みが再利用できる。

5. おわりに

LOTOS実行系を用いたプログラム変換実験の結果について述べた。変換されたプログラムは、生成できるプロセス数に制限があるものの、効率よく動作する。プロセス数の制限は、再帰的動作のループ化により改善される見込みを得た。現在、最適化処理を行う書換系の検討を行っている。書換系を実装した後、変換されたプログラムの実行速度を測定する予定である。最後に、日頃御指導頂くKDD研究所小野所長、浦野次長、通信ソフトウェアグループ小西リーダーに感謝します。

参考文献

[1]: ISO 8807, "LOTOS - A formal description technique based on the temporal ordering of observational behaviour", Feb. 1989.

[2]: 野村, 瀧塚, 長谷川, "LOTOS実行系の実装結果", 情処全大, 1T-9, Mar. 1991.

[3]: 野村, 瀧塚, 長谷川, "LOTOS実行系のための書換え系の設計", 情処全大, 3T-05, Oct. 1991.

[4]: ISO DTR10167, "Guidelines for the Application of Estelle, LOTOS, and SDL", Jan. 1990.