LOTOS実行系における入力仕様からプログラムへの変換

6S-1

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

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により記述された仕様を、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]。本実行系は、仕様をプログラムに変換するトランスレータと、仕様中で規定されるプロセスの実行を管理するスケジューラにより構成される。本稿では、仕様をプログラムに変換するトランスレータの機能について述べる。

2. LOTOS

LOTOSにおいてプロセスの動作およびインタラクションの定義は、CCS[3]を基礎とした記述法を用いる。またデータ構造や値の定義は、抽象データ型(ADT)として代数的に記述される。ここで、インタラクションとはプロセス間での情報の送受のことであり、特にLOTOSではイベントと呼ばれる。

LOTOSによる仕様では、ひとまとまりの動作をプロセスとして記述し、結合オペレータを用いてプロセスの関係を記述する。プロセスは他のプロセスと交換する情報の出入口であるゲートを持ち、その動作はゲートにおいて観測されるイベントの時間的順序を列挙することにより記述される。

3. 入力仕様の変換

3.1 変換の方針

プロセスの実行やイベントによる同期等、LOTOSの言語仕様で規定される基本的な動作は、スケジューラにより実現される[2]。トランスレータは、LOTOSにより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。C言語への変換を検討するにあたり、以下の方針をたてた。

(1) ISOで標準化されたLOTOSの言語仕様に基づいた記述を入力仕様とする。ただし、ADTの難解さおよびその変換処理の複雑さから、ADTを用いたデータ型定義は処理の対象としない。

(2) トランスレータが生成したプログラムに、仕様として記述されない処理等を直接追加することができるよう、元のLOTOS記述が連想できる変換を行う。

(3) スケジューラの提供する関数へ単純に変換するだけでなく、実行効率が高くなるように実行制御オペレータの両辺の記述内容を考慮した変換を行う。

3.2 プログラムへの変換

本実行系の入力となる仕様が、プロセスを単位としてどのように変換されるかを以下に述べる。

図1は、特別なデータ(優先データ)の追越しを考慮したFIFOキューのLOTOSによる仕様[1]の一部である。プロセスNCellは、通常のデータを与えられて生成され、キューの中の一つのセルとして動作する。プロセスNCellは、保持しているデータを出力してプロセスBufferとなるか、優先データが入力されてプロセスXCellとなるかを選択的に実行する。このNCellの変換例を図2に示す。

```
process NCell[inp,outp](x:nrmData):noexit: =
      outp !x ; Buffer[inp,outp]
   [] inp !x ?y:expData ; XCell[inp,outp](y)
endproc
```

図1　LOTOSによる仕様の一部

① プロセス関数

```
Process NCell(local,inp,outp,x)
struct {Descriptor y;} *local;                          /* ㋑ */
Gate *inp, *outp;   Descriptor *x;
{  ProcDescript *PD$Buffer, *PD$XCell;
   if(event(outp,x) != NEXT) {                          /* ㋺-1 */
      PD$Buffer = create(PS$Buffer,inp,outp);
   } else if(event(inp,x,y) != NEXT) {                  /* ㋺-2 */
      PD$XCell = create(PS$XCell,inp,outp,&(local->y));
   } else wait(); }
```

② プロセス構造体

```
ProcStruct PS$NCell = {&NCell,          /*関数アドレス*/
      "NCell",                                /*関数名*/
      NoExit,                           /*noexitな関数*/
      1,                          /*ローカル変数の数*/
      {SORT$expData}};  /*ローカル変数のデータ型*/
```

図2　生成されるプログラム

(1) プロセスの変換

① 仕様に定義されるプロセス

Translation Techniques from LOTOS into C
Shingo NOMURA, Toru HASEGAWA, Takashi TAKIZUKA
KDD Kamifukuoka R & D Laboratories

仕様に記述されたプロセスは、スケジューラが実行を管理するプロセスに対応したCの関数(プロセス関数)に変換(図2①)する。また関数アドレス/プロセス名/ローカル変数等の静的な情報を保持するプロセス構造体の定義(図2②)も同時に生成する。プロセス構造体は、スケジューラがプロセスを生成する時に参照する。

プロセスの内部だけで使用されるローカル変数は、メモリの確保/解放を容易にするためにスケジューラがプロセスごとに割当てて管理する。このためトランスレータは、ローカル変数の定義をプロセス構造体に登録し、割当てられた領域を参照するための構造体宣言を生成する(図2㋑)。

② 仕様に定義されないプロセス

仕様に記述されたプロセスがプロセス関数に変換されるだけでなく、定義された動作からプロセス関数が生成される場合がある。

例えば"g_1 ! x ||| g_2 ? y:int"の記述では、2つの異なるイベントの並列動作を示しており、一つのプロセスでは実行が困難である。このためトランスレータは、並列オペレータの両辺のイベントを独立したプロセス(疑似プロセス)として生成する。

この他にも、オペレータの組合せにより同様に疑似プロセスを生成する場合がある。

(2) イベントの変換

イベントの記述は、スケジューラの提供するevent関数に変換する(図2㋺)。この関数は、要求されたイベントの同期が成立したか、同期が成立しないため次のイベントを要求するかを戻り値として返す。このためイベントの記述は、event関数の戻り値に対応した構造の条件分岐に変換する。例えば、選択実行オペレータ'[]'を用いてイベントが組み合わされている図1の記述を考える。この記述は、一つのイベントの要求を実行(図2㋺-1)し、その戻り値がNEXTであるならば次のイベントの要求を、NEXT以外の戻り値ではイベントが成立した時の処理を実行する構造に変換される。

LOTOSには、イベントの成立に対して制約を付けるガードがある。"[x>0]->g !x ?y:int [y>5]"の記述では、"[x>0]->"および"[y>5]"がガードであり、xの値が正数である時ゲートgに対してイベントが要求され、変数yが5より大きな整数とユニファイ可能な時にイベントが成立する。変数(ここではy)に対するガードは、イベントを要求したプロセスが判定する[2]。このため、もし要求するイベントの変数に対してガードが付けられているならば、ガード条件の判定要求もevent関数の戻り値の一つとなる。

(3) choice文の変換

LOTOSによる仕様を実行可能なプログラムに変換する際に最も問題となるのは、choice文である。

"choice x:int [] [f(x)]->B(x)"は、任意の整定数xのうち、f(x)を満足するものはB(x)が実行できることを意味している。すべての整数xについてf(x)の判定を行い、さらにf(x)を満足するものについてプロセスB(x)を生成して、実行可能性を判定することは現実的でない。上記の記述に対して、例えば"B(x)=g !x ; B'(x)"である時、"g ?x:int [f(x)] ; B'(x)"のようにガードを用いた式に直して変換する。すなわち定数xを変数として扱い、イベントが成立しxが確定した後でf(x)の判定を実行する。

(4) データ型定義

同期の処理とメモリ管理のためにデスクリプタ[2]を介して、データを参照する。デスクリプタ中の識別子によりデータ型を区別する。トランスレータは入力仕様中に宣言されたデータ型名の一覧を出力して、識別子番号を割当てる。

データに対する操作は、Cの関数(操作関数)として実現する。トランスレータは、定義された操作名を元に、操作関数を呼び出すためのextern宣言を生成する。操作関数の実現は、本実行系を用いる実装者が行う。2項演算子は、2引数の関数の呼出しに変換し、関数の戻り値を演算結果とする。論理型演算子は、非零/零の整数値により真偽を返す関数として扱う。

LOTOSは、PASCALのようなスコープルールを持つが、C言語では階層的なスコープを記述できないため、データ型名/操作名はすべてのプロセスから可視であるとした。

4. おわりに

LOTOS実行系の入力仕様から、C言語のプログラムへの変換について検討した。現在、本稿で述べた変換をするトランスレータを作成している。今後、さらに実行効率の高いプログラムが得られるような変換法を検討していく必要がある。最後に、日頃御指導頂くKDD上福岡研究所小野所長、安藤次長、通信ソフトウェア研究室小西室長、若原主任研究員に感謝します。

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,長谷川, 瀧塚,"LOTOS実行系の並列処理環境",情処学会ソフトウェア基礎論研究会,SF.29-4, May, 1989.

[3]:R.Milner, "A Calculus of Communicating Systems", Lecture Notes in Computer Science, Vol.92, Springer-Verlag, 1980.