SDLからLOTOSへの変換

1R−9

安藤津芳*　大友弥生*　更科克幸*　山野敬一郎*　太田正孝*　高橋薫**
*(株)高度通信システム研究所　**東北大学

1. はじめに

仕様を形式的仕様記述言語を用いて定義することで自動プログラミングや検証と言った機械支援を期待すると共に，仕様の曖昧性の除去･欠落の防止を望む傾向が強くなっている．これに伴い通信ソフトウェアの世界でもCCITT, ISOにおいて形式的仕様記述言語が勧告された．しかし，その発生起源の違いからCCITT勧告のSDLは主に交換ソフトウェアで使用され，ISO勧告のEstelle, LOTOSは主にコンピュータソフトウェアで使用されている．また，その言語の特質としてSDL, Estelleは状態遷移モデルを基盤にしているのに対してLOTOSは時系列表記を基盤にし数学的裏付けを重視して仕様検証を追求している．

そこで，本論文ではまず対象とする仕様記述言語の特徴を明確にした後に，交換ソフトウェアのLOTOS的検証の実現を目的として交換ソフトウェア開発に使用されているSDLをLOTOSに変換することを考察する．

2. SDLとLOTOSの特徴

SDLとLOTOSの一般的特徴[1]を表1に示す．

表1. SDLとLOTOSの比較

	SDL	LOTOS
表現形式	意味的に1対1に対応するテキスト表現と図形表現	テキスト表現と図形表現(現在検討中)の二種類
検証性	数学的裏付けは検討中	数学的裏付けがあり検証能力は高い
動作方式	パラレルプロセス	インタリーブプロセス
構造化機能	強力な構造化	見かけ上の構造化
動作モデル	状態遷移モデル	時系列モデル
通信方式	非同期通信に基づき,片方向通信を基本	同期通信に基づき,gateによる多重方向多重通信を基本
時間制御	許容	非許容
データ記述機能	機能の豊富なADT	ADT
共有変数	許容	非許容
適用分野	交換ソフトウェア開発	OSI

3. 変換方針

ここでは，両言語の特徴を加味して，交換ソフトウェア開発の立場から，SDLからLOTOSへの変換を考え，その中でも特に両言語間に差のある構造化機能，動作モデル，通信方式，時間制御を中心にその変換方針を示す．

なお本変換で対象とするSDL仕様は88年度版勧告のBasic SDLを用いて記述され，記述レベルはspecificationを示すものとする．

3.1 構造化機能

SDLの持つ静的な階層構造は，SYSTEM, BLOCK, PROCESSである．一方，LOTOSの持つ階層は，**specification**と**process**で以下**process**の繰り返しである．ここで，SYSTEMと**specification**は対象の仕様全体を示すキーワードである．

このことから，階層構造の対応としては，SDLのBLOCK, PROCESSはLOTOSにおいては総て**process**に変換されることが判る．また，このような対応を付けることは各々のレベルでのスコープの対応もはかれる．そこでこの対応を基本とした変換を考える．

また，SDLの構造定義においてはBLOCK間のインタフェース路であるCHANNEL, PROCESS間のインタフェース路であるSIGNALROUTEが存在し，そしてその中に実際にやり取りされる信号が存在するが，これらをLOTOSに変換するときにどのように扱うかを決める必要がある．LOTOSの**process**間のインタフェース路の定義としては**gate**しか存在しないため，**gate**をSDLのインタフェース路と信号どちらにに対応させるかという判断が必要になるが，ここではインタフェース路に対応させることにした．

これらの方針を適用した静的構造に関する変換例を図1に示す．

```
SYSTEM s1;
  SIGNAL sig1,sig2,sig3;
  CHANNEL c1
    FROM ENV TO b1 WITH sig1;
    FROM b1 TO ENV WITH sig2;
  ENDCHANNEL c1;
  BLOCK b1;
    SIGNALROUTE b1s1
      FROM ENV TO b1p1 WITH sig1;
      FROM b1p1 TO ENV WITH sig2;
    SIGNALROUTE b1s2
      FROM b1p1 TO b1p2 WITH sig1;
      FROM b1p2 TO b1p1 WITH sig2;
    CONNECT c1 AND b1s1;
    PROCESS b1p1 REFERENCED;
    PROCESS b1p2 REFERENCED;
  ENDBLOCK b1;
ENDSYSTEM s1;
```

(a) SDLによる構造定義

```
specification s1[c1]
  behaviour
  hide b1s2 in
    b1[c1,b1s2]
  where
  process b1[c1,b1s2]
    b1p1[c1b1,b1s2]
      |[b1s2]|
    b1p2[b1s2]
    where
    process b1p1[c1,b1s2]
      ...
    endproc
    process b1p2[b1s2]
      ...
    endproc
  endproc
endspec
```

(b) 変換後のLOTOS

図1. SDLからLOTOSへの構造変換例

3.2 動作モデル

先に示したようにSDLは状態遷移モデルであり，LOTOSは時系列モデルである．そこで，変換の際にはLOTOSでの状態遷移モデルんの記述スタイルを明確にする必要がある．LOTOSにおけるこの様な記述スタイルとして次の二通りが考えられる．

(1)SDLのSTATE単位に**process**を定義し，状態遷移を**process**遷移として実現させる方法

(2)SDLのSTATEを管理する状態変数を**process**パラメータとして定義し，STATE単位に**choice**で分割し，状態変数の値で分岐する方法[2]

これらを比較した結果，状態数が多く複雑に絡み合った場合でもわかりやすく，また変換をする場合に状態遷移のためのADTを追加しなくてよい(1)を選択した．

(1)での状態管理の変換例を図2に示す．

3.3 通信方式

SDLが非同期通信を，LOTOSが同期通信を基本にしているため，同期通信による非同期通信の実現が変換において必

Translation from SDL to LOTOS

Tsuyoshi ANDO*, Yayoi OHTOMO*, Katsuyuki SARASHINA*, Keiichirou YAMANO*,
Masataka OHTA*, Kaoru TAKAHASHI**
*Advanced Intelligent Communication System Lab.　**Tohoku University

```
PROCESS p1;
 START;
  NEXTSTATE st1;
 STATE st1;
  INPUT sig1;
   NEXTSTATE st1;
  INPUT sig2;
   NEXTSTATE st2;
 STATE st2;
  INPUT sig1;
   NEXTSTATE st1;
  INPUT sig2;
   STOP;
ENDPROCESS p1;
```

(a) 変換前のSDLの状態遷移

```
process p1[sig1,sig2]:=
 st1
 where
  process st1[sig1,sig2]:=
    sig1;st1[sig1,sig2]
   [] sig2;st2[sig1,sig2]
  endproc
  process st2[sig1,sig2]:=
   sig1;st1[sig1,sig2] [] sig2;exit
  endproc
endproc
```

(b) 状態管理の変換後のLOTOS

図2. SDLの状態遷移のLOTOSへの変換例

須条件となる．そこで，SDLの通信は二者間であり三者以上の同時通信は存在しないという性質を利用することにした．すなわち，各**prosess**は受信専用の**process**を持ち，送信側はその受信用**process**と同期をとるだけで実際の処理を行う**process**とは独立に動けるようにし，処理を行う**process**は信号が必要な場合に受信用**process**と同期通信を行う．

また，SDLにおける通信の実現は送信側の送信路指定(何も指定しない場合を含む)と相手プロセス識別子(PId)指定がある．そこで，これに関しては次の様に考えた．

(1)送信路指定時の変換

この場合は，送信路がそのままLOTOSの**gate**に対応しているので単純な変換で実現可能．

(2)PId指定時の変換

送信専用の**gate**をSYSTEMで唯一作成し，その**gate**への信号送出時の第一パラメータを発信者PId，第二パラメータを送信先PIdとするように変換し，SDLの仮定している通信処理を統括管理・制御する機能を持つ**process**を用意すれば実現は可能だが処理が繁雑になりわかりにくくなる．そこで今回の変換ではこのPId通信を考えない場合に限定した．

3.4 時間制御

SDLの時間制御を，時間概念のないLOTOSで実現する方法として次の二案がある．

(1)SDLの考え方と同様にメタな位置付けでTimer processを定義する．

(2)LOTOSを拡張し時間の概念を取り入れさせる[3]．

今回，LOTOSの拡張を前提とはしていないため，(1)を選択し図3のような Timer meta-processを定義することで，SDLのSET，RESET，TIMEOUTを実現する．

(呼び元)

```
type state is
 sorts state
 opns timer : →state
      normal : →state
      check : state →state
 eqns
    check(timer)
          = timer;
    check(normal)
          = normal;
endtype
process a[gate,T](s:state)
                :=exit
[check(s)=normal] →
  ···
  Timer[T](5)
    |[T]|
  a[gate,T](timer)
[] [check(s)=timer] →
  ( T?sig:time_sig;···
  [] gate;T!Reset;
     ···
  )
[] ···
endproc
```

(Timer meta-process)

```
type time_sig is
 sorts time_sig
 opns Reset,TO : →time_sig
endtype
type Time is NaturalNumber
 sorts Time
 opns 0  :      →Time
    ms  : Nat   →Time
    R_ms : Time  →Nat
    -  : Time   →Time
    _-_ : Time,Nat →Time
 eqns forall a:Nat,b:Time
    ms(a) - a  = 0;
    0 - a      = -(ms(a));
    b - R_ms(b) = 0;
    R_ms(ms(a)) = a;
    ms(R_ms(b)) = b;
endtype
process Timer[T](time_count:Time)
                          :exit:=
  T?t_sig:time_sig [t_sig = Reset];exit
[] [time_count = 0] →T!TO;exit
[] [not(time_cout = 0)] →
          i;Timer[T](time_count - 1)
endproc
```

図3. 概念的なTimer-Meta PROCESS例

3.5 その他

SDLのキーワード単位に変換方針の概略を表2に示す．

表2. 概略変換方針一覧表

キーワード	変換方針
STATE	exitに変換する
DECISION	choiceに変換する．
CREATE	生成processと自processを，必要なgateを同期させてインスタンシエートする．この時，自process起動が再帰となるため，以後の処理を継続するための制御を行う．
SAVE	非同期処理用processとの間の信号のやり取りと，非同期処理用process側でSAVE queueの管理をする
CALL	PROCEDUREもSDLがSTATEを許しているのでprocessとして展開する．従って，変換上はCREATEと同じになる．但し，次処理への接続は，▶ accept out_data in を使用する．
RETURN	exit(out_data)とする．
TASK	処理をprocessとしてPROCEDURE CALLと同様の扱いとする．
LABEL	LABELの部分で処理を分割し，再起動でchoiceする．
JOIN	LABELに対応するchoiceを選択するようにして対象LABELを持つSTATEに対応するprocessを起動する．
VIEW	専用gateを用意し信号受信に対応させる．
REVEAL	専用gateに対する信号送信に対応させる．但し同期による制約から全てのSTATEで展開する．
DCL	LOTOSのprocessのパラメータとして内部データの定義をする．
NEWTYPE	type定義に変換(記述方式単位の変換規則が必要)
GENER-ATOR	type定義をformalに行ない，参照する側のtype定義でactualizedする．
SYNONYM	対象となるデータtypeの追加オペレータとして定義する．

4. まとめ

本論文では，主に交換システムの仕様記述に使用されているSDLで記述された仕様を，LOTOSに変換する方法について考察した．論文中では紙面の都合により両仕様記述言語の特徴的な差異の部分の変換方針を中心に紹介したが，本変換では状態遷移図の変換ができる程度であり，実際の完全な仕様を変換する場合には更に詳細な検討が必要である．また，変換の正当性について触れることができなかったが実際に使えるものを構築する場合には，この証明を行なうことは必須である．そこで，今後は変換の正当性に関しても十分に検討する．

[参考文献]

[1] ISO/IEC:"Guidelines for the Application of Estelle, LOTOS and SDL, "ISO/IEC DTR 10167 (1990)

[2] Chris A. Vissers, et al. :"Architecture and Specification Style in Formal Descriptions of Distributed systems, "Proc. IFIP WG6. 1, 8th Int. Workshop on Protocol Specification, Testing and Verification(North-Holland), pp.189-204 (1988)

[3] Tommaso Bolognesi, et al. :"From Timed Petri Nets TO Timed LOTOS, " Proc. IFIP WG6. 1, 10th Int. Workshop on Protocol Specification, Testing and Verification, pp.377-406 (1990)