通信系の動作記述から各局のプロトコルマシンを生成するための一方法

5E-6

東野輝夫+　木本智久+　谷口健一+　森将豪++
+大阪大学基礎工学部　++滋賀大学

1．はじめに

Bochmannらは文献[1]でLOTOS風言語で記述されたネットワークのサービス仕様から，局の動作列を求める方法を提案している．本稿では，ある制限のもとで，そのようなサービス仕様から，各局のプロトコルマシンを生成する一方法を提案する．文献[1]では，入力となるサービス仕様は，イベントの逐次実行(;)，並列実行(||)，排他的実行(XOR)のみしか許されていなかったが，本稿で提案する方法では，サービス仕様をプロセスとして定義することによって，繰り返しを含むような記述に対しても，各局のプロトコルマシンが導出される．プロトコルマシンは，ある種の決定性プッシュダウンオートマトン(DPDA)の形で与えられる．

2．サービス仕様

サービス仕様は次の6字組で与える．T(SPEC)は開始プロセスps[n]から生成されるイベントの系列集合（一般には無限集合でもよい）全体を表すと定義する．

SPEC＝＜P,E,C,N,S,{ps[n]}＞
- P＝Pm∪Ps
  - Pm：複合プロセス名の集合
  - Ps：単一プロセス名の集合
- E：イベント名の集合
- C：条件名の集合
- N：局の名前の集合
- ps[n]：開始プロセス名（ps∈P，n∈N）
- S：プロセスの集合

但し，p∈Sなる各プロセスは，次の生成規則によって，非終端記号PRから導出されるとする．

[プロセスの生成規則]

```
PR  :== PNm → MP | PNs → EV
PNm :== p[n]   p∈Pm, n∈N
PNs :== p[n]   p∈Ps, n∈N
MP  :== PN ; MP | XOR ( CN , MP , MP ) | PN
PN  :== PNm | PNs
CN  :== cd[n]  cd∈C, n∈N
EV  :== EV ; EV | ( EV || EV ) |
        XOR ( CN , EV , EV ) |
        e[n]   e∈E, n∈N
```
■

以下，MPから導出される系列をプロセス列，EVから導出される系列をイベント列と呼び，プロセスp→αのαをpの本体と呼ぶ．本稿では，プロセスの集合Sは次の3つの条件を満足するものとする．

①p→α∈S且つp→β∈S且つα≠βなる2つのプロセスは存在しない．

②p→α（p∈PNs，αはイベント列）∈Sならば，α中の(e1||e2)なるイベント列e1，e2に対して，
{n|p[n]∈e1}∩{n|p[n]∈e2}＝φ

③p→α（p∈PNs，αはイベント列）∈Sならば，α中のXOR(cd[n],e1,e2)に対して，e1とe2で最初に実行するイベントはcdの条件判定を行う局nで実行される．

[例1]

SPEC＝＜{pm,ps},{a,b,c},{cd},{1,2,3},S,{pm}＞
S＝{pm[1]→XOR(cd[1],ps[1];pm[1],ps[1]),
ps[1]→a[1];(b[2]||b[3]);c[1]}　■

例1は，局1でイベントaを実行した後，局2，3でそれぞれbを並行に実行し，両方の実行が終了した後，局1でcを実行する，という一連の動作を条件cdが偽になるまで繰り返し実行することを表しており，例えば，局1から局2，3にメッセージを送信（放送）するという動作を繰り返す場合に相当する．

3．プロトコルマシン

上のサービス仕様によって，ネットワーク全体のイベントの系列集合T(SPEC)が指定される．T(SPEC)で指定された順にイベントを実行するための各局のプロトコルマシンは次のようなある種のDPDAで与える．

DPDA M＝＜Q,Σ,Γ,E,O,C,δ,{q_0,z_0,i_0}＞
- Q：状態の集合
- Σ：入力記号の集合
- Γ：スタック記号の集合
- E：イベント名の集合
- O：出力記号の集合
- C：条件名の集合
- {q_0,z_0,i_0}：初期状態，初期スタック記号，初期入力記号（図1のバッファの初期値）
- δ：Q×Σ×Γ×C→Q×Γ*×E×O

DPDA Mは，図1のように（自局を含め）すべての局との間に論理的なリンクが存在することを仮定している．また，複数個の局から入力を得た時に一つの状態遷移が行えるよう入力ラインにはバッファがあり，非同期に送られた入力がすべてバッファに入力された時，状態遷移を行う．各入力に対して，Cで指定された条件判定を行い，その条件を満足する時状態遷移を行う．一つの状態遷移によって，E中の一つの

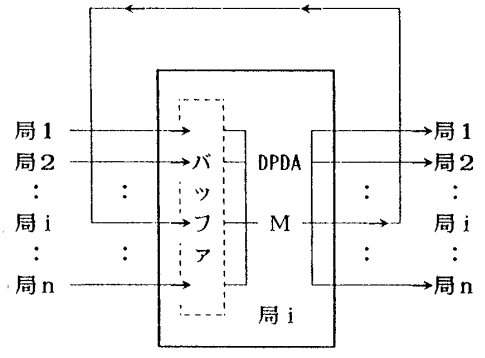

図1　局iのDPDAの概略図

Synthesis of Protocol Machines from Service Specification
Teruo HIGASHINO+, Tomohisa KIMOTO+, Ken'ichi TANIGUCHI+ and Masaaki MORI++
+:Osaka Univ.　++:Shiga Univ.

イベントを実行し，その後，Oで指定された（複数個の）局に同時に出力する．本稿では，次のような集合を指定することによってMの遷移関数δを定義する．

$$\delta=\{\delta(q,\{M(n,\alpha,i)\ldots M(n',\alpha',i)\},z,cd) =<q',z',e,\{M(i,\beta,m)\ldots M(i,\beta',m')\}>\}$$

これは，「状態がq で局nからα,..,局n'からα'を入力し，且つ条件cdを満足する時，イベントe を実行し，その後，状態がq'に遷移し，スタックの内容がz'に変化し，局m にβ,..,局m'にβ' を出力すること」を表している．

4．サービス仕様に対するプロトコルマシン

上の3．で定義したn個（n＝|N|）のDPDAが非同期に動作を行った時に実行されるイベント列の集合が2．で定義したT(SPEC) に一致する時，n個のDPDAはSPECの正しい実現であると定義する．

5．サービス仕様からプロトコルマシンの導出

以下では，サービス仕様SPECからSPECの正しい実現となるプロトコルマシン（DPDA）$M_1,..,M_n$の遷移関数$\delta_1,..,\delta_n$を機械的に導出する方法の概略を述べる．

例えば，2．の［例1］のサービス仕様に対して，次のようなプロトコルマシン（DPDA）M_1,M_2,M_3 が導出されれば，それらのDPDAは例1で指定したサービス仕様の正しい実現になる．

局1(M_1)：$\delta_1=\delta m\cup\delta s$

$\delta m=\{\delta(q,\{M(h,pm[1],1)\},z,true)$
$=<q,z,\phi,\{M(1,*1,1)\}>,$
$\delta(q,\{M(h,*1,1)\},z,cd[1])$
$=<q,z,\phi,\{M(1,ps[1];pm[1],1)\}>,$
$\delta(q,\{M(h,*1,1)\},z,not(cd[1]))$
$=<q,z,\phi,\{M(1,ps[1],1)\}>,$
$\delta(q,\{M(h,ps[1];pm[1],1)\},z,true)$
$=<q,push(z,pm[1]),\phi,\{M(1,ps[1],1)\}>,$
$\delta(q,\{M(h,pop,1)\},z,(\%1\ and\ \%2))$
$=<q,push(pop(z),\alpha),\phi,\{M(1,p[1],1)\}>,$
$\delta(q,\{M(h,pop,1)\},z,(\%1\ and\ not(\%2)))$
$=<q,pop(z),\phi,\{M(1,top(z),k)\}>,$
$\delta(q,\{M(h,pop,1)\},z,(\%3\ and\ \%2))$
$=<q,pop(z),\phi,\{M(1,p[1],1)\}>,$
$\delta(q,\{M(h,pop,1)\},z,(\%3\ and\ not(\%2)))$
$=<q,pop(z),\phi,\{M(1,p[k],k)\}>$ }

但し *1 = XOR(cd[1],ps[1];pm[1],ps[1])
%1 = top(z)=p[k]; α (p∈P)
%2 = k=1
%3 = top(z)=p[k] (p∈P)

$\delta s=\{\delta(q,\{M(1,ps[1],1)\},z,true)$
$=<q_{ps①},z,\phi,\{M(1,①,1)\}>,$
$\delta(q_{ps①},\{M(1,①,1)\},z,true)$
$=<q_{s②},z,a,\{M(1,②,2),M(1,②,3)\}>,$
$\delta(q_{ps②},\{M(2,③,1),M(3,③,1)\},z,true)$
$=<q,z,c,\{M(1,pop,1)\}>$ }

局2(M_2)，局3(M_3) （省略） ■

上のδm では，

Ⓐ局1がプロセス列を入力した時は，プロセス列の2番目以降のプロセスをスタックに格納し，最初のプロセス名を自局に送り，

Ⓑ複合プロセス名を入力した時はその本体を自局に送り，

Ⓒ単一プロセスの終了記号（pop ）を入力した時，スタックトップの最初のプロセス名をポップし，それが自局のプロセスならばそのプロセスを自局に送り，他局のプロセスならばスタックトップの内容をその局に送る．

という操作を記述したものであり，任意の複合プロセスの集合に対してⒶ～Ⓒのような動作を行うδm を導出できる（導出のアルゴリズムは省略）．

また，δs では各イベント列の実行を制御するための状態遷移が記述されている．δs の①～③の記号はそれぞれ

① ② ③
ps[1]→a[1];(b[2]||b[3]);c[1] ---(A)

のように単一プロセスpsの「→」や「;」 に対応する記号である．δs の導出には，文献[1] 同様，図2のように(A) に対する構文木を作成し，次に構文木の各ノードに図2のS,E,..,V,W,Lのような10個の属性を与える．これらの属性を利用すれば，δs は導出できる（属性の与え方や，10個の属性からδs を導出するアルゴリズムは省略）．詳細は別途報告する．

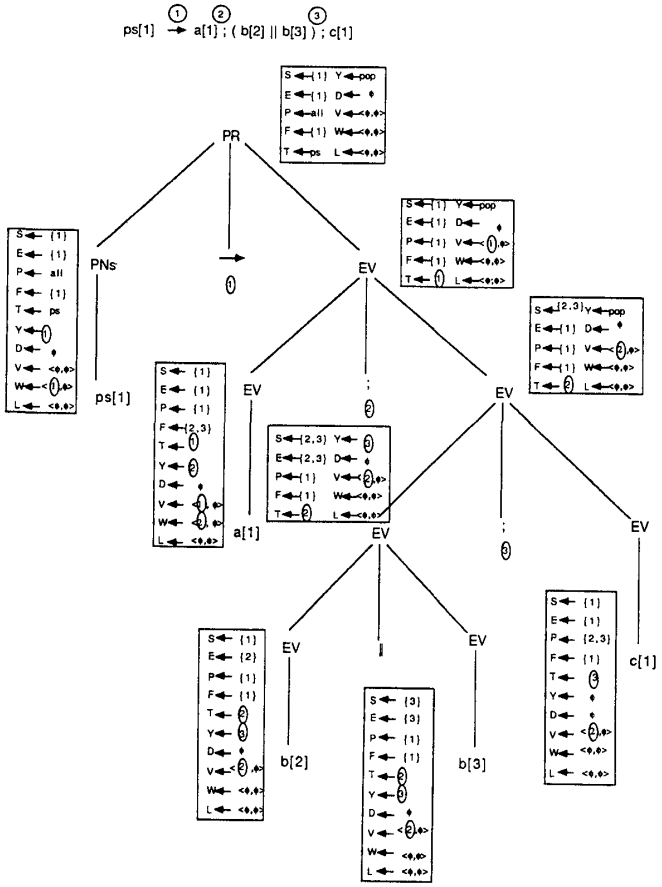

図2 ps[1]→a[1];(b[2]||b[3]);c[1]の構文木と10個の属性

参考文献：Bochmann,G.V. and Gotzhein,R. : Deriving Protocol Specifications from Service Specifications, Proc. of the ACM SIGCOMM'86 (1986).