ソフトウェア基礎論 26－5
（１９８８．１０．１４）

ネットワークのサービス仕様から各局のプロトコルマシンを生成するための一方法

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

本稿では，ネットワークのサービス仕様から，それの正しい実現である各局のプロトコルマシンを機械的に生成するための一方法を提案する．ネットワークのサービスは，各プロトコルマシンが持つプリミティブな機能や動作（以下，イベントと呼ぶ）の実行系列の集合で表されると考える．それらの実行系列（イベント列）の集合を指定するために，LOTOS のサブセットに相当する言語で，イベントの逐次実行，並列実行，排他的実行，繰り返し等が記述できるサービス仕様記述言語を定義した．ネットワーク上のプロトコルマシン群がこの言語で書かれたサービス仕様で指定したイベント列のみを実行する（サービス仕様の正しい実現である）ためには，プロトコルマシン間で同期のためのメッセージの交換が必要である．このようなメッセージの交換を行いながら，イベントを実行する各局のプロトコルマシンの動作表を機械的に導出する．動作表の導出は，属性文法を利用して行う．

Synthesis of Protocol Machines from Service Specification

Teruo HIGASHINO[+], Tomohisa KIMOTO[+],
Ken'ichi TANIGUCHI[+] and Masaaki MORI[++]

[+] Faculty of Engineering Science, Osaka Univ.,
Toyonaka, Osaka, 560 JAPAN
[++] Faculty of Economics, Shiga Univ.,
Hikone, Shiga, 522 JAPAN

This paper presents a method for deriving protocol machines in a network from a service specification of the network. A service specification is written in a LOTOS-like language where sequential executions, parallel executions, alternative executions and their repeats of the events (primitive services of protocol machines) can be described. Protocol machines exchange synchronization messages to control the execution order of the events. The algorithm for deriving the transition function of such protocol machines is described by using an attribute grammar.

1．はじめに

一般に，ネットワークには幾つかの局があり，各局には，それぞれ，プロトコルマシン（計算機や通信制御装置）がある．各プロトコルマシンは，幾つかのプリミティブな機能や動作（例えば，相手局への送信・受信，コネクションの接続・切断等）を実行する装置とみなせる．本稿では，それらの動作（またはそれらの動作とその動作を実行する局名を対にしたもの）を「イベント」と呼び，ネットワークのプロトコルマシン群が実行するイベントを実行順に並べた系列をイベント列と呼ぶ．ネットワーク上の各サービスは単一のイベント列，または，イベント列の集合とみなせるので，これらイベント列の集合をネットワークの「サービス仕様[1]」と呼び，プロトコルマシン群の動作に対する要求を記述したものと考える．すなわち，(1) サービス仕様で指定したイベント列のみをプロトコルマシン群が実行し，且つ，(2) サービス仕様で指定したいかなるイベント列もプロトコルマシン群が実行可能でなければならないことを，要求していると考える．通常，このような要求を満足するためにはプロトコルマシン間で同期のためのメッセージを送受信する必要がある．与えられたサービス仕様に対して，上述の(1)，(2) を満足するために各局のプロトコルマシンがどのように動作しなければならないか，すなわち，(3) どの局からどのようなメッセージを受信した時に，どのイベントを実行して，どのように内部状態を変更し，どの局に対してどのようなメッセージを送信しなければならないか，を記述したものを各局の「プロトコル仕様[1]」と呼ぶ．

本稿では，まず，サービス仕様の一つのクラスを指定し，次に，与えられたサービス仕様に対して，上述の(1)，(2) を満足するような各局のプロトコル仕様を機械的に求める方法を提案する．

近年，サービス仕様を形式的に記述するための言語として，LOTOS[2]が提案され，その標準化が進められている．そこで，本稿では，まず，LOTOS のサブセットに相当する言語を定義し，その言語で書かれたサービス仕様がどのようなイベント列の集合に相当するかを指定する．本稿で定義した言語を用いることによって，イベントの「逐次実行」のみならず，「並列実行（２つのイベント列を並行して実行する）」や「排他的実行（条件判定の結果によって２つのイベント列の何れかを実行する）」，「繰り返し（イベント列を繰り返し実行する）」等が記述できる．

次に，各局のプロトコルマシンのイベント列実行制御を行う部分を，ある種の決定性プッシュダウンオートマトン(DPDA)としてモデル化し，与えられたサービス仕様から各DPDAの動作表を機械的に導出する（DPDAの動作表がプロトコル仕様に相当する）．本稿のサービス仕様の記述のクラスに対して，DPDAは上述の(1)，(2) を満足するプロトコルマシンのモデルとして，最もシンプルなものの一つとして選択した．また，DPDAの動作表の導出には属性文法[3] を利用する．

従来，サービス仕様からプロトコル仕様を合成する方法として，Bochmannらが，イベント，及び「逐次実行」，「並列実行」，「排他的実行」を表すオペレータからなる項（式）から，それぞれの局の動作列（その局のイベントと同期のための局間の通信イベントがどのような順序で実行されなければならないかを記述したもの）を機械的に求める方法を提案している[4][5]．しかし，Bochmannらの方法では，イベントの「繰り返し」等を含むサービスを表現できない等サービス仕様記述言語の表現能力があまり高くない[5]．また，導出された各局の動作列を実現するためのプロトコルマシンのモデルや各プロトコルマシン間で同期のために送受信すべきメッセージの内容等についても触れられていない．これに対して，本稿で提案する方法は，サービス仕様記述言語の表現能力を高め，その実現となるプロトコルマシンの動作表を直接導出している．

2．サービス仕様

以下では，本稿で対象とするサービス仕様の構文とその仕様によって指定されるイベント列の集合を定義する．

［定義2.1］ サービス仕様は次の６字組で定義する．

SPEC＝＜P，E，C，N，S，p[n]＞

P＝Pm∪Ps：プロセス名の集合

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

S：Pに属する各プロセスの定義文の集合

（一つのプロセスに対して一つの定義文）

p[n]：主プロセス名（p∈P，n∈N）

但し，Sに属する各プロセスの定義文は，次の文脈自由文法Gで生成される言語L(G)に属するものとする．

［プロセスの定義文を導出する文脈自由文法G］

G＝（NT，T，R，PR）

NT＝{PR,PNm,MP,PNs,EV,CD}

：非終端記号の集合

T＝{p[n]|p∈P,n∈N}∪{cd[n]|cd∈C,n∈N}∪
{e[n]|e∈E,n∈N}∪{'→','XOR','||',';'}

：終端記号の集合

PR：開始記号

R：生成規則の集合

但し，

R ＝R1∪R2

R1：複合プロセスの定義文を導出する生成規則

R2：単一プロセスの定義文を導出する生成規則

R1＝ {PR :== PNm → MP
PNm :== p[n] p∈Pm, n∈N
MP :== MP ; MP | XOR (CD , MP , MP)
| PNm | PNs
CD :== cd[n] cd∈C, n∈N }

R2＝ {PR :== PNs → EV
PNs :== p[n] p∈Ps, n∈N
CD :== cd[n] cd∈C, n∈N
EV :== EV ; EV | (EV || EV) |
XOR (CD , EV , EV) |
e[n] e∈E, n∈N } ■

各イベント（または，条件，プロセス）e にはそのイベントを実行する局n を付加し，e[n]の形で指定している．また，「$\alpha;\beta$」は，βで定義されるイベント列を実行するためには，αで定義されるイベント列がすべて実行されていなければならないことを表しており，「$\alpha||\beta$」は，αとβで定義されるイベント列を並行して実行してもよいことを表している．「XOR(c[n],α,β)」は，条件c[n]を局nのプロトコルマシンが判定し，真ならばαで，偽ならばβで定義されるイベント列を実行しなければならないことを表している．なお，同一の条件c[n]でも，その条件判定が行われるごとに真偽は異なってもよい．

以下，MPから導出される終端記号列をP項，EVから導出される終端記号列をE項と呼ぶ．各プロセスpに対して，p→α∈Sのαをプロセスpの本体と呼ぶ．単一プロセスpは，その本体（E項）αで指定されたイベントを順次実行することを表している．複合プロセスpは，その本体（P項）α中の各プロセスで指定されたイベント列を，α中に現われるプロセスの順に実行することを表している（「XOR(c[n],α,β)」は，条件c[n]を判定し，その真偽によってP項αあるいはβのいずれかを実行する）．複合プロセスを再帰的に定義する（例えば，p[n]→XOR(c[n],α;p[n],β)）ことによって，αで指定されるイベント列の繰り返しが表現できる．

［定義2.2］

n個のイベントe1,e2,…,en の実行系列をイベント列と呼び，e1･e2･…enで表す．イベント列e（≙e1･e2…en）及びe'（≙e'1･e'2…e'm）に対して，e とe'のイベントをシャッフルした系列（イベント列）集合を(e)#(e')で表す．すなわち，

$(e1\cdot\alpha)\#(e2\cdot\beta)=\{e1\cdot\gamma\mid\gamma\in(\alpha)\#(e2\cdot\beta)\}\cup\{e2\cdot\delta\mid\delta\in(e1\cdot\alpha)\#(\beta)\}$

$(\alpha)\#(NIL)=\{\alpha\}$

$(NIL)\#(\alpha)=\{\alpha\}$ ■

以下，混同がない限り(e)#(e')をe#e'で表す．

［定義2.3］

サービス仕様SPECにおいて，主プロセスp[n]から生成されるイベント列の集合をSPECで指定されるイベント列の集合と呼び，SEQ(SPEC) で表す．

SEQ(SPEC)＝SEQ(p[n])

SEQ(p)＝SEQ(α) （p→α∈Sの時）

SEQ(p1;p2)＝{e1･e2|e1∈SEQ(p1),e2∈SEQ(p2)}

SEQ(XOR(c,p1,p2))＝{c･e1|e1∈SEQ(p1)}∪{(～c)･e2|e2∈SEQ(p2)}

SEQ((p1||p2))＝{e1#e2|e1∈SEQ(p1),e2∈SEQ(p2)} ■

本稿では，条件c の判定を行う場合も，条件判定のためのイベント（条件が真の時はc ，偽の時は(～c)で表す）が実行されると考える．

次に，サービス仕様の例を付す．

［例1］

```
SPEC=<{pm,ps,pe},{p,q,r,s,t},{a,b,c,d},
                              {1,2,3,4},S,pm[1]>
S={ pm[1]→XOR(d[1],ps[1];pm[1],pe[1]) ,
    ps[1]→XOR(a[1],(p[1]||(p[2]||p[3])),
            (XOR(b[2],p[2],q[2])
             ||XOR(c[3],p[3],q[3])));
            (r[2]||s[3])
    pe[1]→t[1] }
```

■

例1のイベントの実行順序をフローチャートのように図式表現すると，図1のようになる．図1の点線で囲まれた（繰り返しを含まない）部分が，単一プロセスpsによって実行されるイベントであり，図1全体で

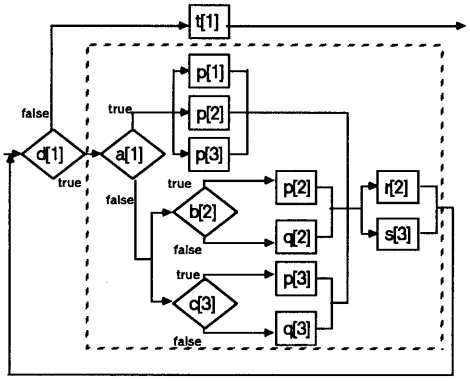


図1　イベントの実行順序

条件d が成り立つ間，psで指定されたイベント列を繰り返し実行し，条件d が成り立たなくなれば，peで指定されたイベントを実行することを表現している．

3．プロトコルマシン

サービス仕様によってネットワーク全体のイベント列の集合SEQ(SPEC) が指定される．サービス仕様SPECで指定した順にイベント列が実行されるためには，ネットワーク上の各局が同期を取りながらイベントを実行する必要がある．本稿では，各局で同期を取ったりイベントを実行するためのプロトコルマシンを次のような状態を持たないDPDAとしてモデル化する．

[定義3.1]

DPDA M＝＜Σ，Γ，E，C，δ，{z_0，f_0}＞

Σ：同期用メッセージ記号の集合

Γ：スタック記号の集合

E：イベント名の集合

C：条件名の集合

z_0：初期スタック記号

f_0：初期メッセージ記号の集合

（図2の ﾊﾞｯﾌｧの初期値）

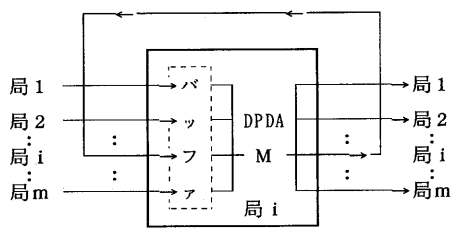


図2　局 i のDPDAの概略図

δ：遷移関数

N：局の名前の集合

N＝{1，2，...，m}　■

DPDA M は，図2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，及び，各局間のデータ転送は誤りがない（誤りが生じた場合には，再送等の機能により，いつかは正しくデータが相手局に受信される）ことを仮定している．各入力ラインにはバッファがあり，受信したデータは，一旦，受信用バッファ（バッファサイズは無限大で，データの到着順序は無関係とする）に蓄えられる．遷移関数δは，(3-1)または(3-2)の何れかの形で指定されると仮定する．

$$\delta(\{MI(s1,\alpha 1),\ldots,MI(sp,\alpha p)\},v) = \langle \omega, e, \{MO(s'1,\beta 1),,MO(s'q,\beta q)\}\rangle \quad --(3\text{-}1)$$

$$\delta(\{MI(s1,\alpha 1),\ldots,MI(sp,\alpha p)\},v) = \text{if } \mu(c) \text{ then } \langle \omega', \{MO(s'1,\beta 1),,MO(s'q,\beta q)\}\rangle \text{ else } \langle \omega'', \{MO(s''1,\gamma 1),,MO(s''r,\gamma r)\}\rangle \quad --(3\text{-}2)$$

(3-1) は，「局s1,...,局spからそれぞれα1,...，αpを入力し，スタックの先頭の記号がv ならば，v を記号列ω∈Γ* に変え，イベントe を実行し，局s'1,,局s'qにそれぞれβ1,,βqを出力すること」を表している．また(3-2) は，「局s1,...,局spからそれぞれα1,...，αpを入力し，スタックの先頭の記号がv ならば，条件c の判定を行う（判定結果が真ならばイベントc を，偽ならば(～c) を実行したとみなす）．条件c が真ならば，スタックの先頭の記号v をω'∈Γ*に変え，局s'1,...,局s'qにそれぞれβ1,...，βqを出力し，条件c が偽ならば，スタックの先頭の記号v をω''∈Γ*に変え，局s''1,...,局s''rにそれぞれγ1,...，γrを出力すること」を表している．尚，μは条件名c を引数として，真または偽を返す関数であり，その条件判定が行われるごとに真偽は異なってもよい．

次に，ネットワーク全体でのプロトコルマシン群の動作について述べる．最初，プロトコルマシン群の中の一つのDPDAの受信用バッファに初期メッセージ記号の集合が与えられるとする．各DPDAにおいて，δで指定された同期用メッセージ記号がすべて受信用バッファに存在し，且つスタックの先頭の記号がそのδで指定されたスタック記号であれば，δで指定された遷移を行う．どの局のDPDAもそれ以上遷移できなくなれば，遷移が終了したと考える．この時，遷移が終了するま

での間に実行されるイベント列が一つ定まり，これを動作列と呼ぶ．初期メッセージ記号の集合πとその集合を与える局n の組$\langle\pi,n\rangle$を一つ定めたとしても，何れかの局が(3-2) のような遷移を行えば，その時点のμ(c) の真偽によって，それ以降にプロトコルマシン群で実行されるイベント列が変化する．また，２つ以上のDPDAで同時に遷移可能となった場合，いずれのイベントが先に実行されるかによって異なる動作列が得られる．このように，πを局n に与えたときに実行される可能性のある動作列の集合を動作列集合と呼び，$\Pi(\langle\pi,n\rangle)$で表す．

［定義3.2］ （サービス仕様とプロトコルマシン群）

サービス仕様SPEC（主プロセスをpm[n] とする）で指定されるイベント列の集合SEQ(SPEC) が，pmを局n の初期メッセージ記号として与えたプロトコルマシン群で実行される動作列集合$\Pi(\langle\{MI(n,pm)\},n\rangle)$に一致する時，それらのプロトコルマシン群はサービス仕様SPECの正しい実現であるという． ■

4．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の正しい実現となるプロトコルマシン（DPDA）$M_1,...,M_m$を機械的に導出する方法について述べる．

4．1　主プロセスが単一プロセスの場合

まず，サービス仕様で指定された主プロセスが単一プロセスである場合について，その正しい実現となるプロトコルマシンの導出方法について述べる．

サービス仕様で指定された順にイベントを実行するためには，各局間で同期用メッセージ記号の送受信を行う必要がある．これらのメッセージ記号は各イベントの実行時，並びに，各条件判定を行うときに受信する必要がある．例えば，［例１］のps[1] が主プロセスの場合（図３参照），図中のp[1]を実行するためには，局１から図中の条件a[1]の判定を済ませたという同期用メッセージ記号を受信する必要がある．また，図中のr[2]やs[3]を実行するためには，次のような５つのうちの一つを受け取る必要がある．

(1) 図中のp[1],p[2],p[3]の実行が終了したことを知らせる同期用メッセージ記号を局1,2,3 からそれぞれ受信（a が真の場合）．

(2) 図中のp[2],p[3] の実行が終了したことを知らせる同期用メッセージ記号を局2,3 からそれぞれ受信（～a･b･c が真の場合）．

(3) 図中のp[2],q[3] の実行が終了したことを知らせる同期用メッセージ記号を局2,3 からそれぞれ受信（～a･b･～c が真の場合）．

(4) 図中のq[2],p[3] の実行が終了したことを知らせる同期用メッセージ記号を局2,3 からそれぞれ受信（～a･～b･c が真の場合）．

(5) 図中のq[2],q[3] の実行が終了したことを知らせる同期用メッセージ記号を局2,3 からそれぞれ受信（～a･～b･～c が真の場合）．

このように，各イベント（あるいは条件判定）を実行するためには，(a) どの局から(b) どのような同期用メッセージ記号を受信すればよいかを定めなければならない．同様に，(c) どの局に(d) どのような同期用メッセージ記号を送信すべきかも定めなければならない．

そこで，まず単一プロセスps及びpsの本体に現れる条件やイベントの出現にそれぞれラベルを付ける．例えば，図３の菱形（条件に相当）や長方形（イベントに相当）の右肩に付けられたアルファベット（A，B，C，...）がpsの本体に現れる条件やイベントの出現に対するラベルの一例である（psの出現に対するラベルはＰで表す）．

本稿では，同期用メッセージ記号を次のように定める．すなわち，イベント（または条件）e1からe2への同期用メッセージ記号は，e1,e2 のラベル（A，Bで表す）の順序対[A，B]を用いる．例えば，上述の(1)の場合，r[2]を実行する局2 が，p[1],p[2],p[3]を実行する局1,2,3 からそれぞれ[D，K]，[E，K]，[F，K] なる同期用メッセージ記号を受信したとき，r[2]の実行が可能となる．尚，最初，局１に単一プロセス名「ps」が初期メッセージ記号として与えられるとし，

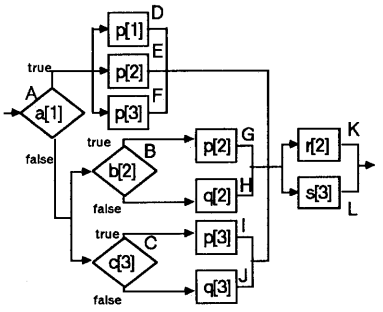


図３　単一プロセスps

局1がpsを受信すれば，a[1]の条件判定を行うために，局1へ[P，A]を送信する．また，最後にr[2]，s[3]を実行する局から局1に送信される同期用メッセージ記号は，それぞれ，[K，P]，[L，P]とし，これら2つの同期用メッセージ記号が共に局1に受信された時，単一プロセスpsで指定されたイベントがすべて終了したとみなす．

各イベント，例えば，図中のr[2]に対して，

{{D，E，F}，{G，I}，{G，J}，{H，I}，{H，J}}　--(4-1)

のようなラベルの集合の族を求めれば，上述の(1)～(5)のような情報，すなわち，r[2]を実行するためには，(a) どの局から(b) どのような同期用メッセージ記号を受信すればよいかが求まる．

同様に，(c) どの局に(d) どのような同期用メッセージ記号を送信すべきかもラベルの集合を求めることによって指定できる．例えば，図中の条件a[1]に対しては，その真偽に応じてメッセージを送るべき局の集合が2通りあるので，

〈{D，E，F}，{B，C}〉　--(4-2)

のようなラベルの集合の対が求まればよい．また，イベントp[1]については，

{K，L}　--(4-3)

のようなラベルの集合を求めれば，(c) や(d) を決めることが出来る．

以下では，(4-1)，(4-2)，(4-3) のようなラベルの集合（またはその族）をどのように機械的に導出するかについて述べる．

以下，E項αの中で最初に実行するイベントの出現の集合をS(α)で表し，最後に実行するイベントの出現の集合の族をE(α)で表す．例えば，図3の単一プロセスps[1]の本体中に現われる「XOR(a[1]，(p[1]||(p[2]||p[3]))，(XOR(b[2]，p[2]，q[2])||XOR(c[3]，p[3]，q[3])))」及び「(r[2]||s[3])」をそれぞれα1，α3とすれば，(4-1)，(4-3)はそれぞれE(α1)，S(α3)に相当する．

α；βなるE項において，S(β) に属するイベントe[n]はE(α)に属するイベントを実行する局から同期用メッセージ記号を受信すればよく，これをP(e[n])で表す．同様にE(α)に属するイベントe[n]はS(β)に属するイベントを実行する局へ同期用メッセージ記号を送信すればよく，これをF(e[n])で表す．例えば，(4-1)，(4-2)，(4-3) はそれぞれP(r[2])，F(a[1])，P(p[1])に相当する．上述の(a)～(d)を求めるためには，単一プロセスの本体に現れるいくつかのイベントやE項に対してS，E，P，F の値を計算すればよい．本稿では，文献(4) でBochmannらが提案した属性文法を利用する方法を応用してS，E，P，F の値を計算する．そのために，まず2.で述べたプロセスを生成する文脈自由文法Gを用いて，与えられた単一プロセスの定義文 ps[n]→αの構文木tを作成する．この構文木tの葉以外の各ノードに対して上述の4個の属性[3] S，E，P，F を付加する．

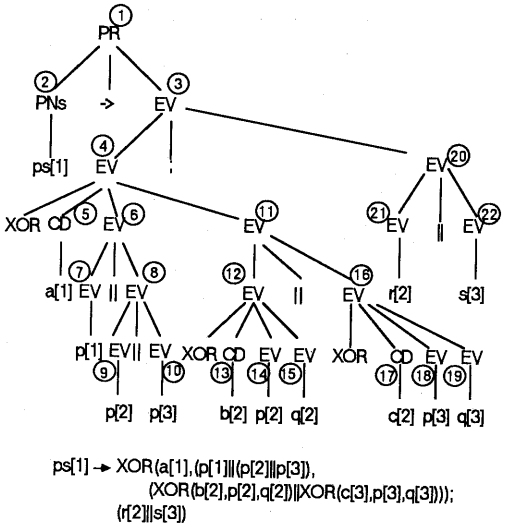
［属性の指定］

プロセスの定義文を生成する文脈自由文法Gに対して，次のような属性文法[3] G_A=（G，A_t，S_R）を定義する．また，単一プロセスの定義文ps[n]→α の構文木tに対して，G_A を用いて作成された属性つき

表1　属性値　S，E，F，P　に対する意味規則

生成規則	属性 S	属性 E
PR :=PNs→EV	S(PR) :=S(PNs)	E(PR) :=E(EV)
PNs:=p[n]	S(PNs):={Ψ(p[n])}	E(PNs):={{Ψ(p[n])}}
EV:=EV_1;EV_2	S(EV) :=S(EV_1)	E(EV) :=E(EV_2)
EV:=EV_1||EV_2	S(EV) :=S(EV_1)∪S(EV_2)	E(EV) :={{α，β}| {α}∈E(EV_1)，{β}∈E(EV_2)}
EV:=XOR(CD，EV_1，EV_2)	S(EV) :=S(CD)	E(EV) :=E(EV_1)∪E(EV_2)
EV:=e[n]	S(EV) :={Ψ(e[n])}	E(EV) :={{Ψ(e[n])}}
CD:=c[n]	S(CD) :={Ψ(c[n])}	E(CD) :={{Ψ(c[n])}}
Ψ(e[n]) Φ	イベント（または条件名）e[n]に付けられたラベル Φ(Ψ(e[n])) :=n　(e[n]に付けられたラベルΨ(e[n]))からそのイベントを実行する局n を計算する関数)	

生成規則	属性 P	属性 F
	P(PR) :=ϕ	F(PR) :=S(PR)
PR :=PNs→EV	P(PNs):=P(PR) P(EV) :=E(PNs)	F(PNs):=S(EV) F(EV) :=F(PR)
EV:=EV_1;EV_2	P(EV_1):=P(EV) P(EV_2):=E(EV_1)	F(EV_1):=S(EV_2) F(EV_2):=F(EV)
EV:=EV_1||EV_2	P(EV_1):=P(EV) P(EV_2):=P(EV)	F(EV_1):=F(EV) F(EV_2):=F(EV)
EV:=XOR(CD，EV_1，EV_2)	P(CD) :=P(EV) P(EV_1):=E(CD) P(EV_2):=E(CD)	F(CD) :=<F(EV_1)，F(EV_2)> F(EV_1):=F(EV) F(EV_2):=F(EV)



ノード	属性 S	属性 E	属性 P	属性 F
①	{P}	{{K, L}}	ϕ	①−S
②	{P}	{{A}}	ϕ	③−S
③	{A}	{{K, L}}	②−E	①−S
④	{A}	{{D, E, F}, {G, I}, {G, J}, {H, I}, {H, J}}	②−E	⑳−S
⑤	{A}	{{A}}	②−E	⟨⑥−S, ⑪−S⟩
⑥	{D, E, F}	{{D, E, F}}	⑤−E	⑳−S
⑦	{D}	{{D}}	⑤−E	⑳−S
⑧	{E, F}	{{E, F}}	⑤−E	⑳−S
⑨	{E}	{{E}}	⑤−E	⑳−S
⑩	{F}	{{F}}	⑤−E	⑳−S
⑪	{B, C}	{{G, I}, {G, J}, {H, I}, {H, J}}	⑤−E	⑳−S
⑫	{B}	{{G}, {H}}	⑤−E	⑳−S
⑬	{B}	{{B}}	⑤−E	⟨⑭−S, ⑮−S⟩
⑭	{G}	{{G}}	⑬−E	⑳−S
⑮	{H}	{{H}}	⑬−E	⑳−S
⑯	{C}	{{I}, {J}}	⑤−E	⑳−S
⑰	{C}	{{C}}	⑤−E	⟨⑱−S, ⑲−S⟩
⑱	{I}	{{I}}	⑰−E	⑳−S
⑲	{J}	{{J}}	⑰−E	⑳−S
⑳	{K, L}	{{K, L}}	④−E	①−S
㉑	{K}	{{K}}	④−E	①−S
㉒	{L}	{{L}}	④−E	①−S

図4　属性つき構文木

構文木(属性つき解析木)[3] をΨ(t)で表す.

G_A=（G, A_t, S_R）

G ：プロセスの定義文を生成する文脈自由文法

A_t：属性の集合

$A_t = SA_t \cup IA_t$

SA_t={S, E}：合成属性[3]

IA_t={P, F}：相続属性[3]

S_R ：表1の意味規則[3]の集合. ■

尚，表1中の補助関数Ψ(e[n])は，イベントe[n]に付加されたラベル（Aで表す）を表す．逆に，Aとラベル付けされたイベントに対し，Φ(A)でそのイベントを実行する局を表す．

［属性値の計算方法］

4個の属性値は，次の順序で計算すれば容易に求めることができる．

(1) 合成属性S， E

(2) 相続属性P， F ■

図4に図3の単一プロセスの定義文ps[1]→α に対する属性つき構文木を付す．尚，構文木中には直接属性値を表記せず，各ノードに①〜⑫の番号を付け，各番号ごとに4つの属性値を右側の表中にまとめて表記している．ノード③の属性Pを「②−E」と指定しているが，これはノード②の属性Eと同じ属性値{{A}}を持つことを表している．同様に，ノード⑤の属性Fを「⟨⑥−S, ⑪−S⟩」と指定しているが，これはノード⑤の属性Fの値が⟨{D, E, F}, {B, C}⟩であることを表している．

次に，上述の4つの属性を用いて，各局のプロトコルマシンの遷移関数を指定する方法について述べる．先にも述べたように，単一プロセスで指定されたイベントを実行するためには，

(イ) プロセスの開始時

(ロ) 各条件の判定時

(ハ) イベントの開始時

(ニ) プロセスの終了時

に，それぞれ同期を取る必要がある．(イ) については，図4のノード②のようにそのノードの子供が単一プロセス名であるようなノードの属性を用いて決定できる．同様に，(ロ) については，ノード⑤のようにそのノードの子供が条件名であるようなノードの属性を用いて，(ハ) については，ノード⑦のようにそのノードの子供がイベント名であるようなノードの属性を用いて，(ニ) については，ノード①のような構文木の根の属性を用いて求めることができる．また，各ノードの属性Pの値を用いて同期用メッセージ記号を受信する局やそのメッセージ記号の内容を指定できる．同様に，同期用メッセージ記号を送信する局やそのメッセージ記号の内容についてはそのノードの属性Fを用いて求めることができる．すなわち，局k のプロトコルマシンの遷移関数δ_k を次のように指定すればよい．

[プロトコルマシンの遷移関数の指定]

次の(イ)～(ニ)で指定したδ_k の和集合を局k のプロトコルマシンの動作表と呼び，局k はこの動作表に従って遷移を行えばよい．

(イ) 単一プロセス名ps[k] を子供に持つノードPNs に対して

$\{\delta_k(\{MI(k,ps)\},z)$

$=\langle z,\phi,\{MO(\Phi(q),[\Psi(ps[k]),q])\mid q\in F(PNs)\}\rangle\}$

(ロ) 条件名をc[k]を子供に持つノードEVに対して

$\{\delta_k(\{MI(\Phi(m),[m,\Psi(c[k])])\mid m\in P'\},z)$

=if $\mu(c)$

then

$\langle z,\{MO(\Phi(q),[\Psi(c[k]),q])\mid\langle q,r\rangle\in F(CD)\}\rangle$

else

$\langle z,\{MO(\Phi(r),[\Psi(c[k]),r])\mid\langle q,r\rangle\in F(CD)\}\rangle$

$\mid P'\in P(CD)\}$

(ハ) イベント名をe[k]を子供に持つノードCDに対して

$\{\delta_k(\{MI(\Phi(m),[m,\Psi(e[k])])\mid m\in P'\},z)$

$=\langle z,e,\{MO(\Phi(q),[\Psi(e[k]),q])\mid q\in F(EV)\}\rangle$

$\mid P'\in P(EV)\}$

(ニ) 単一プロセス名をps[k] のとき，ps[k]→α の構文木の根を表すノードPRに対して

$\{\delta_k(\{MI(\Phi(m),[m,m'])\mid m\in E',m'\in S(PR)\},z)$

=if $z\neq z_\emptyset$ then

$\langle NIL,\phi,\{MO(first(z),z)\}\rangle \mid E'\in E(PR)\}$

(但し，first(α) はP項α 中の最初のプロセスを実行する局名を表す補助関数．

first($\alpha 1;\alpha 2$) := first($\alpha 1$)

first(XOR(cd[n], $\alpha 1$, $\alpha 2$)) := n) ■

[サービス仕様の正しい実現となるプロトコルマシン]

主プロセスが単一プロセスps[n] であるようなサービス仕様SPECに対して，SPECの正しい実現となる各局k のプロトコルマシンM_k ($1\leq k\leq m, m=|N|$) を次のように構成する．但し，ps[n] の定義文ps[n]→α の構文木をt とする．また，ps[n] 及びα 中の条件やイベントに付加されたラベルの集合をΔ で表す．

$M_k=\langle\Sigma,\Gamma,E,C,\delta_k,\{z_\emptyset,f_{k\emptyset}\}\rangle$

Σ = $\{ps\}\cup\{[\omega 1,\omega 2]\mid\omega 1,\omega 2\in\Delta\}$

Γ = $\{z_\emptyset\}$

E = α に含まれるイベント名の集合

C = α に含まれる条件名の集合

δ_k = 上の(イ)～(ニ)で定義した動作表

$z_\emptyset$: M_k の初期スタック記号

$f_{k\emptyset}$: 受信用バッファの初期メッセージ記号の集合

$f_{k\emptyset}$ = if k=n then $\{MI(k,ps)\}$ else ϕ ■

例えば，図3のps[1] の場合，図4の属性つき構文木から局1の遷移関数δ_1 として，

$\delta_1=\{\delta(\{MI(1,ps)\},z)$

$=\langle z,\phi,\{MO(1,[P,A])\}\rangle$ $\}\cup$

$\{\delta(\{MI(1,[P,A])\},z)$

=if $\mu(a)$

then

$\langle z,\phi,\{MO(1,[A,D]),MO(2,[A,E]),$

$MO(3,[A,F])\}\rangle$

else

$\langle z,\phi,\{MO(2,[A,B]),MO(3,[A,C])\}\rangle\}\cup$

$\{\delta(\{MI(1,[A,D])\},z)$

$=\langle z,p,\{MO(2,[D,K]),MO(3,[D,L])\}\rangle\}$

が導出される．

4．2 主プロセスが複合プロセスの場合

以下では，[例1] のpmのように，サービス仕様の主プロセスが複合プロセスとして指定されている場合について，その正しい実現となるようなプロトコルマシン（DPDA）の組$M_1,\ldots,M_n$ を与える．

複合プロセスは，通常幾つかの単一プロセスを順次実行するように記述されている．各単一プロセスの実行を開始するためには，直前の単一プロセスが終了したことを知らせるメッセージを受け取る必要がある．このようなプロセスの実行順の管理方法としては，ネットワーク上の一つの特別な局が管理する方法（集中管理方式）とそのような局を設けずにネットワーク上

の各局がそれぞれのローカル情報のみを用いて管理する方法（分散管理方式）がある．本稿では，このうち，分散管理方式によって，与えられたサービス仕様の実現となるようなプロトコルマシン(DPDA)を生成する方法を提案する．

提案する方法では，複合プロセス名を受信した時は，その本体（P項）αを送信し，P項αを受信した時は，そのうちの最初のプロセス名以外のP項をスタックに保存し，最初のプロセス名をそのプロセスを実行する局に送信する．その操作を繰り返し，ある局が単一プロセスを名受信すれば，4.1 で述べたような同期用メッセージ記号の送受信を行いながらその単一プロセスで指定されたイベント列を実行する，という方法を用いる．すなわち，提案する方法に於ける各局のDPDAの動作の概略は次の通りである．

［複合プロセスに対する動作］

Ⓐ主プロセスがpm[k] である場合，局kの初期メッセージ記号の集合としてMI(k,pm[k]) を，局k以外の局の初期メッセージ記号の集合にはϕを与える．
最初，局kがpm[k] を受信すれば，pmの本体(P項)αの実行後，局kで終了処理のための特別なプロセスend[j]を実行せよというメッセージα;end[k] を局first(α) に送る．

Ⓑ局kがP項pm[k];end[j]（pmは複合プロセス名）を入力した時は，
pmの本体(P項)αの実行後，局jで終了処理のプロセスend[j]を実行せよ，というメッセージα;end[j]を局first(α) に送る．

Ⓒ局kがP項 pm[k];β（pm は複合プロセス名，$\beta \neq$ end[j]）を入力した時は，
βをスタックにプッシュし，pm[k] の本体（P項）αの実行後，end[k]を実行せよ，というメッセージα;end[k] を局first(α) に送る．

Ⓓ局kがP項XOR(cd[k],β,γ);θ を入力した時は，条件cdの真偽を判定し，
真ならば，β;θ を局first(β) に送り，
偽ならば，γ;θ を局first(γ) に送る．

Ⓔ局kがP項ps[k];β（psは単一プロセス名）を入力した時は，βをスタックにプッシュし，局k自身に単一プロセス名psを送る．

Ⓕ局kが単一プロセス名psを入力した時は，4.1 で述べたような遷移を行う．

Ⓖ局kがend[k] を入力した時は，
スタックの先頭の要素（P項）αをポップし，αを局first(α) に送る．但し，スタックが空（スタックの先頭が初期スタック記号z_0）ならば，動作を終了する． ■

例えば，［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（遷移関数，初期入力記号列以外は省略）．

局1 (M_1)：

$\delta_1 = \delta m \cup \delta s$

$\delta m = \{\delta(\{MI(1,pm)\},z)$
$=<z,\phi,\{MO(1,*1)\}>$,
$\delta(\{MI(m1,pm[1];end[1])\},z)$
$=<z,\phi,\{MO(1,*1)\}>$,
$\delta(\{MI(m1,*1)\},z)$
$=$if $\mu(d)$
then
$<z,\phi,\{MO(1,ps[1];pm[1];end[1])\}>$,
else
$<z,\phi,\{MO(1,pe[1];end[1])\}>$,
$\delta(\{MI(m1,ps[1];pm[1];end[1])\},z)$
$=<z\cdot pm[1];end[1],\phi,\{MO(1,ps)\}>$,
$\delta(\{MI(m1,pe[1];end[1])\},z)$
$=<z\cdot end[1],\phi,\{MO(1,pe)\}>$,
$\delta(\{MI(m1,end[1])\},z)$
$=$if $z \neq z_0$ then
$<NIL,\phi,\{MO(first(z),z)\}>$ | $m1 \in N$ }

但し *1 = XOR(d[1],ps[1];pm[1],pe[1]);end[1]

δs ：サービス仕様中に現われるすべての単一プロセス（psやpe）に対して，4.1 で述べたような遷移関数の動作表の集合を作成し，その和集合をδs とする．但し，サービス仕様中に現われる任意の2つの単一プロセス$p_1 \rightarrow \alpha_1$，$p_2 \rightarrow \alpha_2$ のイベントや条件の出現にはすべて相異なるラベルを付けるものとする．

初期入力記号列＝{MI(1,pm[1])}

局2 (M_2)，局3 (M_3)　（省略） ■

詳細については省略するが，4.1 同様，属性文法を用いて，上述のような遷移関数δ_1（$=\delta m \cup \delta s$）も機械的に導出できる．よって，2.で述べたサービス仕様の実現となるプロトコルマシン（DPDA）の組$M_1, \ldots, M_n$ が機械的に求まる．

5．おわりに

本稿では，サービス仕様のクラスを定義し，次に与えられたサービス仕様の実現となるプロトコルマシン（DPDA）の組$M_1,\ldots,M_n$を機械的に求める方法を提案した．尚，実際のプログラム等では，本稿で導入した[A，B]の形の同期用メッセージ記号は，例えば一つの整数値として表すのが自然であろう．等価なプロトコルマシン群で，用いる同期用メッセージ記号数がなるべく少ないものを求めることは理論的に興味ある問題であるが，むずかしい問題と思われる．

参考文献

(1) Bochmann, G.V. : "Concepts for Distributed Systems Design", Springer-Verlag (1983).
水野，井手口訳：”分散処理システムデザイン”，工学社 (1987).

(2) ISO/TC97/SC21/WG16-1 N299 : Lotos - A formal Description Technique. または
Brinksma, E. : "A Tutorial on LOTOS", Proc. of the IFIP WG 6.1 Workshop (Protocol Specification, Testing and Verification V), North-Holland, pp.171-194 (1985).

(3) 佐々政孝：”属性文法”，コンピュータソフトウェア，Vol.3，No.4，pp.73-91 (1986).

(4) Bochmann, G.V. and Gotzhein, R. : "Deriving Protocol Specifications from Service Specifications", Proc. of the ACM SIGCOMM'86, pp.148-156 (1986).

(5) 伊藤，市川：”通信分野における自動プログラミング”，情報処理，Vol.28,No.10，pp.1405-1411 (1987).