マルチメディア通信と分散処理　52－39
（1991．9．27）

LOTOSに基づいたプロトコル仕様の導出

馬渕博之†,高橋薫††,白鳥則郎†

†東北大学工学部,††東北大学電気通信研究所

あらまし　大規模ソフトウェアなどを複数人で協同で設計する場合、通常、大規模仕様を小規模な仕様に段階的に分割することにより開発が進む。つまり、もとの仕様Pはn個の仕様P1からPnに分割される。このように、仕様全体の意味を保存しながら、1つの仕様をこれと等価な複数の仕様に分割することを仕様の分割問題と呼ぶ。本論文では、仕様記述言語LOTOSを用い、もとの仕様と弱バイシミュレーション等価であるような分割のアルゴリズムを与える。分割アルゴリズムでは、仕様のモデルとして、同期モデルと非同期モデルを扱っている。もとの仕様として、通信システムのサービス仕様を考えると、本論文で与える分割アルゴリズムは、このサービス仕様をネットワーク上の複数サイトで、プロトコルとして実現する、系統的な方法であると捉えることができる。つまり、与えられたサービス定義から、それと矛盾なく実現するプロトコル仕様を導出する基盤を成すアルゴリズムとなっている。

Method for deriving Protocol specification from Service Definition

Hiroyuki Mabuchi,Kaoru Takahashi,Norio Shiratori

†Faculty of Engineering ,Tohoku University

††Research Institute of Electrical Communication ,Tohoku University

†Aoba,Aramaki-aza,Aoba-ku,Sendai-shi,980 Japan

††Katahira 2-1-1,Aoba,Aoba-ku,Sendai-shi,980 Japan

Abstract　This paper presents an algorithm for decomposition of an original LOTOS specification into a number of subprocesses.They interact via either a synchronous or an asynchronous communication channel.The algorithm guarantees that the original specification and the decomposed subprocesses are weak bisimilar.This decomposition algorithm can be applied to the stepwise design of large communication software,e.g. the refinement of a service definition into a protocol specification.

1 まえがき

大規模ソフトウェアなどを複数人で協同で設計する場合、通常、大規模仕様を小規模な仕様に段階的に分割することにより開発が進む。つまり、図1のように、もとの仕様Pはn個の仕様P1からPnに分割される。このように、仕様全体の意味を保存しながら、1つの仕様をこれと等価な複数の仕様に分割することを、仕様の分割問題と呼ぶ。

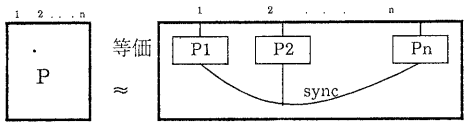


図-1 仕様の分割

仕様の分割に関する従来の研究[1]では、仕様記述言語LOTOSを用いて、分割数を2に限定した場合について、分割アルゴリズムを構成している。文献[2]では非形式的な言語でサービス仕様を記述した場合、プロトコル仕様を導出するアルゴリズムについて検討しているが、これらの仕様間の等価性については、言及していない。本論文では、仕様記述言語LOTOSを用い、文献[1]のように分割数を2に限定しないで、もとの仕様と弱バイシミュレーション等価であるような分割のアルゴリズムを与える。

もとの仕様を、通信システムをblack boxと見なすサービス仕様と考えると、本論文で与える分割アルゴリズムは、ネットワーク上の複数サイトでそのサービスをプロトコルとして実現する、ひとつの系統的な方法であると捉えることができる。つまり、与えられたサービス定義から、それと矛盾なく実現するプロトコル仕様を導出する基盤を成すアルゴリズムとなっている。

従来の研究と本論文との比較を表-1に示す。

文献 / 項目	[1]	[2]	本論文
記述言語	LOTOS (形式的)	BNF記法	LOTOS (形式的)
分割数	2個	n個	n個
等価性の基準	弱バイシミュレーション	なし	弱バイシミュレーション
サブプロセスにまたがる選択	あり	なし	あり
並列オペレータの有無	なし	あり	あり

表-1 本論文の位置付け

2 LOTOSに基づいた仕様の分割法

2.1 仕様の記述モデル

2.1.1 仕様の記述構文

仕様および分割された仕様は、次表で示すLOTOSのsubsetで記述されるものとする。ここでもとの仕様は、表-2の構文-Aおよび構文-Bで与えられ、分割された仕様は、表-2の構文-Aと構文-Bと構文-Cとで表される。なお、文献[1]では、構文-Aのみがもとの仕様として許されている。また、省略記法として、choiceおよびparallel-compositionが2つ以上続くときには、次表の記法を用いる。

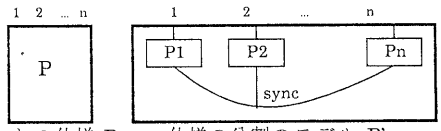
[仕様の構文A]	
inaction	**stop**
choice	B1[]B2
action prefix	g;B i;B
instantiation	P
[仕様の構文B]	
parallel-composition	B1|||B2
[仕様の構文C]	
action prefix	gd;B with d of the form !E
parallel-composition	B1|[g1,...,gn]|B2
hiding	**hide** g1,...,gn **in** B
[省略記法]	
Σ{Bi|i∈I}	2つ以上のchoice
|||{Bi|i∈L}	2つ以上のparallel-composition

表-2 仕様の構文および省略記法

3.1.2 分割された仕様のモデル

(1)同期モデル

図2-1に示す通り、もとの仕様は、同期の場合、同期ゲートsyncを介して相互作用するような、サブプロセスに分割される。



もとの仕様 P ・ 仕様の分割のモデル P'

図 2-1 仕様の分割のモデル(同期)

(2)非同期モデル

非同期の場合は、図2-2に示す通り、もとの仕様は、ある通信媒体を介して相互作用するような、サブプロセスに分割される。

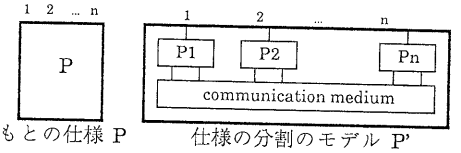


図 2-2 仕様の分割のモデル(非同期)

2.2 分割法の基本的な考え方

基本的な考え方は次の2ステップより成る。
[step 1] アクション集合の分割
[step 2] サブプロセスの構成

step 1で、もとの仕様Pが与えられたとき、機能を表す観測可能アクションの集合を人間がヒューリスティックに分割する。step 2では、その分割に基づいて、アルゴリズムが、もとの仕様と弱バイシミュレーション等価となるように、もとの仕様をサブプロセスP1からPnに分割する。まず、同期の場合について考察する。

2.2.1 同期の場合

同期の場合、もとの仕様Pと仕様の分割のモデルP'間の弱バイシミュレーション等価(≈で示す)をLOTOSを用いて定式化すると、次式となる。

$P \approx P'$

すなわち、

$P \approx$ (hide sync in P1|[sync]|...|[sync]|Pn)

次に、上述の分割法step 1 とstep 2 を詳細化すると、次のようになる。

[step 1] アクション集合の分割

プロセスPの観測可能アクションの集合を、Act(P)で表す。このとき、次の2つをヒューリスティックに行う。

① プロセスPが与えられているときに、$Act(P) \subseteq A$ であるようなAを構成する。

ここで、分割の対象となる、集合Aを構成する。

② Aに対して、$A1 \cup A2 \ldots \cup An = A$ かつ $A1 \cap A2 \ldots \cap An = \varnothing$ であるA1,..., AnをAの分割と呼ぶ。これは、人間により与えられる。

次にAに対して、分割されたサブプロセスのアクション集合にあたる、A1,..., Anまでを与える。

[step 2] サブプロセスの構成

図 2-1 において、$P \approx P'$ が成立するには、次の2つの性質を満たすP1,P2,...,Pnを構成すればよい。

①$Act(P1) \cap A \subseteq A1$, $Act(P2) \cap A \subseteq A2$,..$Act(Pn) \cap A \subseteq An$

構成される各PiがAiの範囲内で動作し、その他に制御用として、その他のPj($i \neq j$)と協調動作しても良いことを示している。

② hide sync in P1|[sync]|P2|...|[sync]|Pn≈P

式②はもとの仕様と、分割された仕様との等価性を示している。

①と②を満足する、P1,P2,...,Pnを、次の写像T1からTnで構成する。具体的には、次章で与える。

T1(P)=P1 , T2(P)=P2 , ... , Tn(P)=Pn

2.2.2 非同期の場合

非同期の場合については、3.2で説明する。

3.分割アルゴリズム

step 1 は人間がヒューリスティックに与える。本章では、step 2 のサブプロセスを構成するためのアルゴリズム(分割アルゴリズム)を、同期と非同期の場合に分けて与える。

3.1 同期の場合

同期の場合、満たすべき条件は式①と②により、次のとおりである。

$Act(T1(P)) \cap A \subseteq A1, \ldots, Act(Tn(P)) \cap A \subseteq An$
hide sync in T1(P)|[sync]|...|[sync]|Tn(P) ≈ P

3.1.1 アルゴリズムで用いる記法

①同期アクションは次のように表される。
sync!m (m∈M)
ただし、Mはメッセージの有限集合とする。

②各メッセージを区別するために、次のものを用いる。
メッセージ変数m_i(iは有限のインデックス集合上のi)

③分割されたアクションの集合A1,...,Anの各要素を次のように表される。

$a_i \in A1$, $b_j \in A2$, ...

3.1.2 アルゴリズムの概要

提案するアルゴリズムによって、導出される各サブプロセスは、互いに完全に独立しているわけではない。そこで、もとの仕様と分割された仕様間の弱バイシミュレーション関係が成り立つように、不要なアクションは削除し、各アクションの後には、適当な同期アクションを付加する。この同期アクションは、syncゲート上のメッセージの同期により実現する。

3.1.3 アルゴリズム

アルゴリズムは、次のように制限を1つずつ取り除いていく形で、4つのstageで構成する。この制限を次の表-3 で与える。なお、以下、記号Bは仕様を表す動作式とする。

	Ⅰ	Ⅱ	Ⅲ
stage 1	無	無	無
stage 2	有	無	無
stage 3	有	有	無
stage 4	有	有	有

Ⅰ.2つ以上のサブプロセスにまたがる選択

Ⅱ.内部イベント

Ⅲ.並列オペレータ

表-3 stageごとの制限

3.1.1.1 stage 1のアルゴリズム

最もきつい制限を持つ場合の、アルゴリズムは次のように与えられる。

サブプロセスPs(1≦s≦n)におけるイベント$x_i(i\in I_s)$を$x_{s,i}$と表す。

【 Algorithm 1】

If　　$B=\Sigma\{x_{s,i};B_i|i\in I_s\}$
Then　　$Ts(B)=\Sigma\{x_{s,i};sync!m_i;Ts(B_i)|i\in I_s\}$
　　$Tt(B)=\Sigma\{sync!m_i;Tt(B_i)|i\in I_s\}$ $(t\neq s)$

『定理1』

T1からTnをAlgorithm1で定義する。このとき

hide sync **in** T1(B)|[sync]|...|[sync]|Tn(B) ≈ B　　□

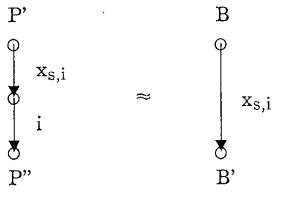
[証明]

$B=\Sigma\{x_{s,i};B_i|i\in I_s\}$とする。エクスパンション[3],[4]により、

hide sync in T1(B)|[sync]|...|[sync]|Tn(B)=

$\Sigma\{x_{s,i};i;$(hide sync in T1(Bi)|[sync]|...|[sync]|Tn(Bi))$|i\in I_s\}$

P'=hide sync in T1(B)|[sync]|...|[sync]|Tn(B)
$B=\Sigma\{x_{s,i};B_i|i\in I_s\}$ とすると、



P",B'は、P',Bと同じ形になる。これより、状態間の関係がとれる。よって、

{<hide sync in T1(B)|[sync]|...|[sync]|Tn(B),B>|Bはstage 1 プロセス}
∪{<i;hide sync in T1(B)|[sync]|...|[sync]|Tn(B),B>|B はstage 1プロセス}

は、弱バイシミュレーション関係である。　　□

3.1.1.2 stage 2のアルゴリズム

2つ以上のサブプロセスにまたがる選択を許す場合のアルゴリズムは、次のとおりである。

【 Algorithm 2】

If　$B=\Sigma\{x_{1,i};Bi|i\in I_1\}$
　　⋮
　　$[]\Sigma\{x_{n,i};Bi|i\in I_n\}$

Then　$T1(B)=\Sigma\{x_{1,i};sync!mi;T1(Bi)|i\in I_1\}$
　　$[]sync!poll;(\Sigma\{sync!mi;T1(Bi)|i\in I_2\}$
　　$[]sync!poll;\{$
　　⋮
　　$\Sigma\{sync!mi;T1(Bi)|i\in I_n\}$
　　$[]sync!poll;T1(B)$
　　$\}$
　　⋮
$Tn(B)=\Sigma\{sync!mi;T1(Bi)|i\in I_1\}$
　　$[]sync!poll;(\Sigma\{sync!mi;T1(Bi)|i\in I_2\}$
　　$[]sync!poll;\{$
　　⋮
　　$\Sigma\{x_{n,i};sync!mi;T1(Bi)|i\in I_n\}$
　　$[]sync!poll;T1(B)$
　　$\}$

『補題1』

X1からXnまでをプロセスとすると、Pを次のように定義する。

P=X1[]i;(X2[]i;(.....Xn[]i;P))

そうすると、P ≈ X1[]...[]Xn である。

[証明]

略

『定理2』

T1からTnをAlgorithm2で定義する。このとき

hide sync **in** T1(B)|[sync]|...|[sync]|Tn(B) ≈ B

[証明]

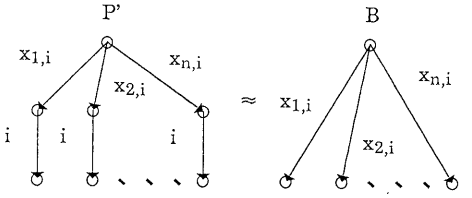
エクスパンションにより

hide sync **in** T1(B)|[sync]|...|[sync]|Tn(B) = P

$P=\Sigma\{x_{1,i};\mathbf{i};($**hide** sync
　　in T1(Bi)|[sync]|...|[sync]|Tn(Bi))$|i\in I_1\}$
　　$[]i;(\Sigma\{x_{2,i};\mathbf{i};$**hide** sync
　　in T1(Bi)|[sync]|...|[sync]|Tn(Bi) $|i\in I_2\}$
　　$[]i;\{$
　　⋮
　　$[]i;(\Sigma\{x^{n}_{i};\mathbf{i};$**hide** sync
　　in T1(Bi)|[sync]|...|[sync]|Tn(Bi) $|i\in I_n\}$
　　$[]i;P$　　$)$

補題1より、P ≈ P'

$P' = \Sigma\{x_{1,i};i;(\text{hide sync in}T1(Bi)|[sync]|...|[sync]|Tn(Bi))|i\in I_1\}$
⋮
$[]\Sigma\{x_{n,i};i;(\text{hide sync in}T1(Bi)|[sync]|...|[sync]|Tn(Bi))|i\in I_n\}$



図より状態間の関係を全てとることができる。よって、

{<hide sync in T1(B)|[sync]|...|[sync]|Tn(B),B>|B はstage 2プロセス}
∪{<i;hide sync in T1(B)|[sync]|...|[sync]|Tn(B),B>|B はstage 2プロセス}

は、弱バイシミュレーション関係

3.1.1.3 stage 3のアルゴリズム

内部イベントの記述を許すばあいのアルゴリズムは、次のとおりである。

【Algorithm 3】

If $B = \Sigma\{x_{1,i};Bi|i\in I_1\}$ ... $[]\Sigma\{x_{n,i};Bi|i\in I_n\}$ $[]\Sigma\{i;Bk|k\in K\}$
Then $T1(B) = \Sigma\{x_{1,i};sync!mi;T1(Bi)|i\in I_1\}$
$[]\Sigma\{i;sync!mk;T1(Bk)|k\in K_1\}$
$[]sync!poll;(\Sigma\{sync!mi;T1(Bi)|i\in I_2\}$
$[]\Sigma\{sync!mk;T1(Bk)|k\in K_2\}$
$[]sync!poll;\{$
⋮
$\Sigma\{sync!mi;T1(Bi)|i\in I_n\}$
$[]\Sigma\{sync!mk;T1(Bk)|k\in K_n\}$
$[]sync!poll;T1(B)\ )$
⋮
$Tn(B) = \Sigma\{sync!mi;T1(Bi)|i\in I_1\}$
$[]\Sigma\{sync!mk;T1(Bk)|k\in K_1\}$
$[]sync!poll;(\Sigma\{sync!mi;T1(Bi)|i\in I_2\}$
$[]\Sigma\{sync!mk;T1(Bk)|k\in K_2\}$
$[]sync!poll;\{$
⋮
$\Sigma\{x_{n,i};sync!mi;T1(Bi)|i\in I_n\}$
$[]\Sigma\{i;sync!mk;T1(Bk)|k\in K_n\}$
$[]sync!poll;T1(B)\ )$

『定理3』

T1からTnをAlgorithm3で定義する。このとき、

hide sync in T1(B)|[sync]|...|[sync]|Tn(B) ≈ B

[証明]

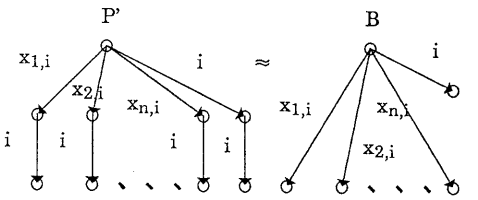
エクスパンションにより

hide sync in T1(B)|[sync]|...|[sync]|Tn(B) = P

$P = \Sigma\{x_{1,i};i;(\text{hide sync in } T1(Bi)|[sync]|...|[sync]|Tn(Bi))|i\in I_1\}$
$[]\Sigma\{i;i;(\text{hide sync in } T1(Bk)|[sync]|...|[sync]|Tn(Bk))|k\in K_1\}$
$[]i;(\Sigma\{x_{2,i};i;\text{hide sync in } T1(Bi)|[sync]|...|[sync]|Tn(Bi)\ )|i\in I_2\}$
$[]i;\{$
⋮
$[]i;(\Sigma\{x_{n,i};i;\text{hide sync in } T1(Bi)|[sync]|...|[sync]|Tn(Bi)\ )|i\in I_n\}$
$[]\Sigma\{i;i;(\text{hide sync in } T1(Bk)|[sync]|...|[sync]|Tn(Bk))|k\in K_n\}$
$[]i;P$
)

補題1より、P ≈ P'

$P' = \Sigma\{x_{1,i};i;(\text{hide sync in } T1(Bi)|[sync]|...|[sync]|Tn(Bi))|i\in I_1\}$
⋮
$[]\Sigma\{x_{n,i};i;(\text{hide sync in } T1(Bi)|[sync]|...|[sync]|Tn(Bi))|i\in I_n\}$
$\Sigma\{i;i;(\text{hide sync in } T1(Bk)|[sync]|...|[sync]|Tn(Bk))|k\in K_1\}$
⋮
$\Sigma\{i;i;(\text{hide sync in } T1(Bk)|[sync]|...|[sync]|Tn(Bk))|k\in K_n\}$



図より状態間の関係を全てとることができる。よって、

{<hide sync in T1(B)|[sync]|...|[sync]|Tn(B),B>|Bはstage 3プロセス}
∪{<i;hide sync in T1(B)|[sync]|...|[sync]|Tn(B),B>|B はstage 3プロセス}

は、弱バイシミュレーション関係 □

3.1.4 stage 4のアルゴリズム

全ての制限のない場合のアルゴリズムは、次のとおりである。

【Algorithm 4】

$B_l = \Sigma\{g_i;B_i|i\in I_l\}$
$B = |||\{B_l|l\in L\}$ $L=\{1,2,...,n\}$

とすると、Algorithm 3のTs(B)を用いて、

$Ts(B) = |||\{Ts(B_l)|l\in L\}$

『定理4』

T1からTnをAlgorithm 4で定義する。このとき、

hide sync in T1(B)|[sync]|...|[sync]|Tn(B) ≈ B

[証明]

$B_l = \Sigma\{g_i;B_i|i\in I_l\}$

$B = |||\{B_l|l\in L\}$ $L=\{1,2,...,n\}$
$Ts(B) = |||\{Ts(B_l)|l\in L\}$

まず、L=2の場合について証明する。

つまり、hide sync T1(B)|[sync]|T2(B) ≈ B with B=B1|||B2を証明する。

前の定理により、

B1 ≈ hide sync T1(B1)|[sync]|T2(B1)

B2 ≈ hide sync T1(B2)[[sync]]T2(B2)

このとき、

B1|||B2 ≈ hide sync T1(B1)[[sync]]T2(B1) ||| hide sync T1(B2)[[sync]]T2(B2)

文献[4]により、

hide sync T1(B1|||B2)[[sync]]T2(B1|||B2)

=hide sync
(T1(B1)|||T2(B2))[[sync]](T1(B1)|||T2(B2)))

=hide sync
(T1(B1)[[sync]]T2(B1))|||(T1(B2)[[sync]]T2(B2)

よって、

B ≈ hide sync T1(B)[[sync]]T2(B)

弱バイシミュレーション関係が成り立つ。

同様に、hide sync T1(B)[[sync]]...[[sync]]Tn(B) ≈ B

も成り立つ。 □

3.2.非同期の場合

非同期の場合は、図2-2で表したような、各エンティティが通信媒体で結合したものを考える。アルゴリズムの本質は、同期信号sync!mを送信send!mと受信rec!mにわけることである。媒体では、送信信号を受けとったら、各エンティティ全てに受信信号を送る操作を行う。このとき、仕様の分割のモデルP'を定式化すると、次のようになる。

Mはメッセージのユニバースとする。このときmediumは、

Medium=Med1|||...|||Medn
where
Med1=Σ{send1!m;(rec2!m;...;recn!m);Med1|m∈M}
⋮
Medn=Σ{sendn!m;(rec1!m;...;rec(n-1)!m;Medn|m∈M}

よって、P'は、

hide send1,rec1,...,sendn,recn in
(T1(B)|||...|||Tn(B))[[send1,rec1,...,sendn,recn]]Medium

である。

ここで、次の2つの条件を満たすアルゴリズムを構成する。

Act(T1(P))∩A⊆A1,...,Act(Tn(P))∩A⊆An
P ≈ P'

ただし、アルゴリズムの詳細およびその証明については紙幅の都合により省略する。

4 アルゴリズムの適用例

本節では、2で与えた仕様の分割アルゴリズムの適用例を以下に与える。

[適用例]

B=a;stop[]b;stop|||a;b;stop のとき、

A={a,b}に対して、A1={a},A2={b}となるように分割を与え、アルゴリズムを適用した結果は以下のように導出される。

T1(B)=a;sync!ma1;stop
[]sync!poll;(sync!mb1;stop
[]sync!poll;T1(B))
|||a;sync!ma2;sync!poll;T1(B)

T2(B)=sync!ma1;stop
[]sync!poll;(b;sync!mb1;stop
[]sync!poll;T2(B))
|||sync!ma2;b;sync!mb2;stop

5.むすび

本論文では、与えられたLOTOS仕様から、それをn個(n≧2)のサブプロセス仕様に分割するアルゴリズムを提案した。同時に、分割されたサブプロセス仕様の合成が、もとの仕様と弱バイシミュレーション等価であることを示した。

現在、このアルゴリズムをワークステーション上でユーザに支援するソフトウェア環境を構成中である。

参考文献

[1]R.Langerak,"Decomposition of functionality:a correctness preserving LOTOS transformation," Tenth International IFIP WG 6.1 Symposium on Protocol Specification,Testing andVerification,ed.L.Logrippo,R.L.Probert,H.Ural,(1990),pp. 203-218.

[2]R.Gotzhein,G.v.Bochmann,Deriving protocol specifications from service specifications:an extended algorithm,to be published in ACM TOPLAS 1990.

[3]ISO,"LOTOS - a formal description technique based on the temporal ordering of observational behaviour - ,"ISO 8807(1989).

[4]R.Milner,Communication and Concurrency,Prentice-Hall International(1989)

[5]高橋,神長,白鳥,LOTOS言語の特質と処理系の現状と動向,情報処理,Vol.31,No.1,pp.35-46(1990).

[6]VISSERS,C.A. and LOGRIPPO,L. The importance of the service concept in the design of data communications protocols.in:M.Diaz(ed.),Protocol Specification,Testing,and Verification,V,Proc.of the IFIP WG 6.1 Workshop,Toulouse-Moissac,France,June 10-13,1985,North-Holland,Amsterdam 1986,pp.3-17