等価性に基づく LOTOS 仕様の記述スタイル変換法

郷 健太郎† 白鳥則郎†

LOTOS は ISO で標準化された分散システムのための FDT(形式記述技法)である．LOTOS は数学的基盤を持つ反面，記述性や理解性に乏しい．そこで共通の記述スタイルを導入しようという研究が行われている．LOTOS の代表的な記述スタイルは，記述対象とするシステムに対して，異なった仕様記述の構造を基にしている．したがって，等価性に基づく記述スタイル変換法が，LOTOS を用いたシステム開発工程において，重要な役割を示す．本論文では，LOTOS の代表的な記述スタイルのうち，モノリシックスタイルから制約指向スタイルへの等価性に基づく記述スタイル変換法を構成し，その諸性質を導出する．本論文の手法は，ソフトウェア工学の分野において，以下の応用に有効となる．(1)大規模仕様を小規模仕様群に細分する，(2)詳細仕様の再構造化，(3)仕様のモジュール化．

Correctness Preserving LOTOS Transformation from the Monolithic Style to the Constraint-oriented Style

KENTARO GOH† and NORIO SHIRATORI†

LOTOS is a FDT (Formal Description Technique) for distributed and concurrent systems developed by ISO. LOTOS has a mathematical background, but has poor describability and understandability. Thus, a study of introduction of common LOTOS specification style is necessary. In LOTOS, the specification styles are based on a different architectual models. The correctness preserving transformations therefore play an important role in the LOTOS based development strategy. In this paper, we propose a correctness preserving transformation from the monolithic style to the constraint-oriented style, which are representative of the specification styles in LOTOS, and make the properties of the transformation clear. The proposed transformation is useful to the following applications: (1) decomposition of a large specification into small specifications, (2) restructuring of complex specifications and (3) modulalization.

1. まえがき

LOTOS[1]は，情報通信システムや分散システムの仕様を曖昧なく，厳密に記述するために ISO によって開発された仕様記述言語の一つである．LOTOS は等価性の概念に基づいた数学的基盤を持つ反面，記述性や理解性に乏しい．これを解決するために，共通の記述スタイルを構築しようという研究が行われている[2),3)]．共通の記述スタイルを導入することで，多人数の仕様記述者が分担して作業する場合に，①仕様全体の同質性を保ち，②仕様の記述性，理解性を向上させることが可能になる．LOTOS の代表的な記述スタイルには次の4つがある[3)]．

(1) モノリシックスタイル (monolithic style),
(2) 制約指向スタイル (constraint-oriented style),
(3) 資源指向スタイル (resource-oriented style),
(4) 状態指向スタイル (state-oriented style).

これらの記述スタイルは，記述対象とするシステムに対して，異なった仕様記述の構造を基にしている．したがって，等価性に基づく記述スタイル間の変換法は，ソフトウェア工学の面で以下のような応用分野を持つ．

(a) 大規模仕様を小規模仕様に細分する，
(b) 仕様を実システムの機能にあうように分割する，
(c) サービス仕様からプロトコル仕様を導出する，
(d) 詳細仕様の再構造化，
(e) 仕様のモジュール化．

等価性に基づく LOTOS の記述スタイル変換法に関する従来の研究には，次の3つがある．

(A) (2), (3), (4)から(1)への変換法[1)]，
(B) (1)から(3)への変換法[4),5)]，
(C) (2)から(3)と，(3)から(2)への変換法[6)]．

(A)はシステムの動作を明確化する上で，LTS (Labelled Transition System[1),7)]) を導出するために必要

† 東北大学工学部情報工学科
Department of Information Engineering, Faculty of Engineering, Tohoku University

な変換法である．(B)は応用として(a)，(b)，(c)に適し，(C)は(b)，(d)に適する変換法である．

文献9)，10)，11)は並列オペレータを導入することにより，変換されるプロセスの表現能力を向上させているが，基本的には(B)の変換法に属する．

本論文では，従来提案されていなかった，(a)，(d)，(e)の応用に有効となる，(1)のモノリシックスタイルから(2)の制約指向スタイルへの変換法を構成し，その諸性質を導出する．

以下，2章でモデルの定式化と諸定義を与え，3章でLOTOSのモノリシックスタイルから制約指向スタイルへの変換アルゴリズムを与える．4章で本手法を適用した例を示し，5章では考察を行う．

なお，以下本論文ではLOTOSの知識を前提として議論を進めていく．LOTOSの詳細については文献1)を参照されたい．

2. 準　　備

2.1 モノリシックスタイルと制約指向スタイル

モノリシックスタイルと制約指向スタイルは，ともにシステムの内部構造を記述しない，サービス定義に適した記述スタイルである．

特に前者では，システムを1つのかたまりとして考え，外部から観測可能なアクションを記述する．

一方，後者では，システムをアクションの生起可能性に関する個別的な制約の集まりとして表現する．具体的には，あるゲートに関与するアクションの時間順序をLocal-constraintのプロセス（以下，LCと略記）として記述し，別々のゲートにおいて生じる意味的に関連し合ったアクションの生起順序をGlobal-constraintのプロセス（以下，GCと略記）として記述する．各LCは独立であり，LC全体とGC全体が同期をとるように記述する．

2.2 仕様記述言語

本論文では仕様記述言語として，LOTOSのサブセットを用いる．具体的には，以下の6つの構文を用いる．

(1) 無動作 **stop**

(2) アクションプレフィックス $a;P$

(3) チョイス $P_1[\ ]P_2$

(4) 並列合成

(4-1) 一般並列 $P_1|[a_1,\cdots,a_n]|P_2$

(4-2) 独立並列 $P_1|||P_2$

(4-3) 完全同期並列 $P_1||P_2$

ただし，P, P_1, P_2は(1)から(4)の構文を用いて表現されるプロセスであり，$a, a_1, \cdots, a_n$は内部アクション**i**を含まない外部観測可能アクションである．

変換される仕様は(1)から(3)の構文を用いて記述され，変換後の仕様は(1)から(4)の構文を用いて表現される．

本論文ではプロセスに関して，以下の記法を用いる．

[記法1]（チョイスの一般形）記号Σとプロセスの集合Sを用い，ΣSと書くことにより，S中の要素間のチョイスを表す．$\Sigma\emptyset$は**stop**を意味する．□

例えば，$I=\{1,2,3,4\}$とした時，$\Sigma\{P_i|i\geq 2, i\in I\}$は，$P_2[]P_3[]P_4$を意味する．

[記法2]（プロセス間の構文的等価性）$\equiv$はプロセス上の関係であり，$P_1\equiv P_2$によりプロセスP_1とP_2が構文的に等しいことを表す．□

[記法3]（プロセスに記述されているアクションの明示）プロセスPに記述されているすべての観測可能アクションの集合がAであり，これを明示的に表現したいとき$P[A]$と書く．□

[記法4]（プロセスを構成するアクション集合）プロセスPの展開過程で発生可能な観測可能アクション全体を$Act(P)$で表す．□

ここでプロセスの展開過程とは，与えられたプロセスに文献1)で定義された公理と推論規則を順次適用していって LTS を導出する過程のことをいう．例えば，プロセス $a;(b;\textbf{stop}\ []\ c;\textbf{stop})$ において LTS を導出すると**図1**のようになる．このプロセスの展開過程で発生可能なアクション全体は $\{a,b,c\}$ であり，展開過程で生じ得るすべてのプロセスは，$a;(b;\textbf{stop}\ []\ c;\textbf{stop})$ と $b;\textbf{stop}\ []\ c;\textbf{stop}$ と2つの **stop** である．

2.3 記述スタイル変換問題の定式化

モノリシックスタイルの定義を基に，本論文では，変換されるプロセスを次の形式で表現する．

[定義1]（モノリシックスタイルの一般形）

$P\equiv\Sigma\{a_i;P_i|i\in I\}$

ここで，Iは有限の添字集合であり，P_iはPと同

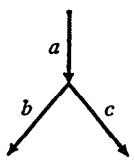


図1 Labelled Transition System
Fig. 1 Labelled Transition System.

じ形式を持つプロセスである. □

本論文では，変換されるプロセスに対して，以下の条件を与える.

[入力条件A] (変換されるプロセスとそのアクション集合に関する条件)

(1) 変換されるプロセスPは [定義1] のものとする.

(2) $Act(P)$ を n 分割したアクション集合を $A_1, A_2, \cdots, A_n$ とする. ただし $A_1 \cap A_2 \cap \cdots \cap A_n = Act(P)$ かつ $A_i \cap A_j = \emptyset$ $(i \neq j)$. □

このとき，変換問題を次のように定式化する.

[定義2] (モノリシックスタイルから制約指向スタイルへの記述スタイル変換問題) [入力条件A] が与えられたとき，次の2つの性質を満たす LC : $LC_1, LC_2, \cdots, LC_n$ と GC : $GC_1, GC_2, \cdots, GC_m$ $(m \geq 1)$ を構成する.

(1) $Act(LC_1) = A_1$, $Act(LC_2) = A_2, \cdots, Act(LC_n) = A_n$.

(2) $P \cong (LC_1 ||| LC_2 ||| \cdots ||| LC_n) |[G]| (GC_1 ||| GC_2 ||| \cdots ||| GC_m)$. ただし $\{G\} \subseteq Act(P)$ であり，$\cong$ はある同値関係である. □

なお本論文では，同値関係として〜，つまり強 bisimulation 等価[7]を用いる.

議論を簡単にするため，以下のような制限を与える.

[制限B]

(1) LC の数を $n=2$ とする. つまり，LC_1 と LC_2 を対象とする.

(2) 内部アクション i は用いない.

(3) プロセスPにおいて記述されている各アクションは，正確に1回だけ記述中に出現する.

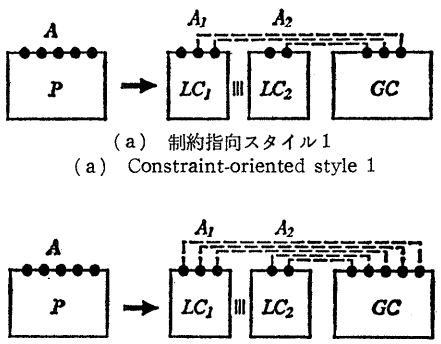
(4) プロセス P の表現に再帰は用いない. □

本論文では，[定義2] と [制限B] の下で，それぞれ以下の2種類の記述スタイルに変換する2つのアルゴリズムを提案する.

[制約指向スタイル1] (文献2)の記述スタイル，**図2(a)**) $P \sim (LC_1 ||| LC_2) |[G]| (GC_1 ||| GC_2 ||| \cdots ||| GC_m)$ ただし，$\{G\} = \cup \{Act(GC_k) | k \in K\}$, $K = \{1, 2, \cdots, m\}$ である.

[制約指向スタイル2] (文献3)の記述スタイル，図2(b)) $P \sim (LC_1 ||| LC_2) || (GC_1 ||| GC_2 ||| \cdots ||| GC_m)$

(注) || (完全同期並列) は |[G]| (一般並列) の特別な場合であるから，制約指向スタイル2は制約指向スタイル1の特別な場合である.



(a) 制約指向スタイル1
(a) Constraint-oriented style 1

(b) 制約指向スタイル2
(b) Constraint-oriented style 2

図2 変換問題のモデル
Fig. 2 Models of the transformation problem.

2.4 諸定義

以下に諸定義を与える.

[定義3] (部分シーケンス) B をアクション集合とする. あるアクションシーケンス $t, s \in B^*$ に対して，

$$t = r_1 \cdot s \cdot r_2$$

であるような $r_1, r_2 \in B^*$ が存在するとき，s は t の部分シーケンスであると言う. ここで $a \cdot b$ はアクションシーケンス a の後ろへのアクションシーケンス b の連接を表す. □

[定義4] (交互シーケンス) B_i と B_j を互いに素なアクション集合とする. あるアクションシーケンス $s \in (B_i \cup B_j)^*$ の長さが2以上であり，かつ s において B_i と B_j のアクションが交互に並んでいる場合，s は B_i と B_j に関する交互シーケンスであると言う. □

[定義5] (トレース集合) プロセス P のトレース集合 $Tr(P)$ を次のように定義する.

$$Tr(P) = \{s \in Act(P)^* | \text{ ある } P' \text{ が存在して } P = s \Rightarrow P'\}$$

ここで，$P = s \Rightarrow P'$ は，P が観測可能なアクションのシーケンス s により P' に遷移可能であることを示す. s 中の各アクションの前後には任意個の内部アクションがさしはさまれて良い. □

[定義6] (サブトレース集合) プロセス P のトレース集合 $Tr(P)$ のサブトレース集合 $STr(P)$ を次のように定義する.

$$STr(P) = \{s \in Act(P)^* | \text{ あるプロセス } P', P'' \text{ および } a \in Act(P) \text{ が存在して } P = s \Rightarrow P' - a \rightarrow P''\}$$

ここで，$P' - a \rightarrow P''$ は，P' が観測可能なアクション

a の生起により P'' に遷移可能であることを示す. □

[定義7] (最大トレース集合) プロセス P の最大トレース集合 $MTr(P)$ を次のように定義する.

$$MTr(P)=Tr(P)-STr(P)$$ □

[定義8] (プレアクション) P, P' をプロセスとする.

$$P=s\Rightarrow P''-a\rightarrow P'$$

であるような $s\in Act(P)^*$ と $a\in Act(P)$ とプロセス P'' が存在する時, a を P における P' のプレアクションと言い, $Pre_P(P')$ と書く. P が明らかな場合, $Pre(P')$ と書く. □

[定義9] (イニシャル) あるプロセス P を展開したときに, 直接生起させることができるようなアクションの集合を, P のイニシャルといい, $Init(P)$ と書く. □

例えば, $a;b;P_1[\ ]c;d;P_2[\ ]e;f;P_3$ のイニシャルは, $\{a,c,e\}$ である.

[定義10] (連結プロセス) あるプロセス P を展開する過程において生じる任意のプロセス P' とアクション集合 B に対し, 次の2つの条件のいずれかが成立するとき, P' をアクション集合 B による連結プロセスという.

(1) $Pre(P')$ が存在せず, かつすべての $a\in Init(P')$ に対して $a\in B$ であるとき.

(2) $Pre(P')\in B$ であり, かつすべての $a\subset Init(P')$ に対して $a\in B$ であるとき. □

[定義11] (切断プロセス) あるプロセス P を展開する過程において生じる任意のプロセス P' と, 互いに素なアクション集合 B_i と B_j に対し, 次の2つの条件のいずれかが成立するとき, P' をアクション集合 B_i と B_j による切断プロセスという.

(1) $Pre(P')$ が存在せず, かつある $a\in Init(P')$ が存在して $a\in B_i$ であり, かつある $b\in Init(P')$ が存在して $b\in B_j$ であるとき.

(2) $Pre(P')\in B_i$ であり, かつある $a\in Init(P')$ が存在して $a\in B_j$ であるとき. □

3. モノリシックスタイルから制約指向スタイルへの変換アルゴリズム

本節では本論として, モノリシックスタイルから対応する制約指向スタイルへの変換アルゴリズムを構成する. まず準備として, 変換されるプロセス P から, あるアクション集合 B の要素だけで構成されるプロセスを切り出すために, 写像 $Rest$ を導入する.

[定義12] (写像 $Rest$) モノリシックスタイルのプロセス $P\equiv\sum\{a_i;P_i|i\in I\}$ とアクション集合 B に対して, 写像 $Rest(P,B)$ を次のように定義する.

$$Rest(P,B)\equiv\sum\{a_i;\ Rest(P_i,B)|a_i\in B,i\in I\}\ []\sum\{Rest(P_i,B)|a_i\notin B,i\in I\}$$

ただし, $Rest(\mathbf{stop},B)\equiv\mathbf{stop}$. □

3.1 制約指向スタイル1への変換アルゴリズム

プロセスの再現性とプロセス間の並列合成関係を考慮し, 写像 $Rest$ を用いて以下の戦略を構成する.

[戦略1]

(1) **LC の構成法**

P から $Rest$ を用いて A_1 と A_2 だけで構成されるプロセスに写像したものを, それぞれ LC_1 と LC_2 とする.

(2) **連結プロセスと切断プロセスに関する構成法**

P を展開する過程におけるすべてのプロセスに対して,

(a) A_1 と A_2 による連結プロセスを LC_1 か, あるいは LC_2 に再現する (これは(1)の構成法によって再現される).

(b) A_1 と A_2 による切断プロセスを GC のうちのどれか1つに再現する.

ここで再現とは, 同じ名前のプレアクションと同じイニシャルを持つようなプロセスを構成することをいう. プレアクションが存在しないプロセスについては, 同じイニシャルを持つようなプロセスを構成することをいう. 例えば, **図3** (a)のプロセス P において a_1 が生じた後のプロセス P' は, プレアクション a_1 とイニシャル $\{a_2,a_3\}$ を持つ. 図3(b)のプロセス LC_1 において a_1 が生じた後のプロセス LC_1' は, P' のものと同じ名前のプレアクション a_1 を持ち, 同じイニシャル $\{a_2,a_3\}$ を持つ. したがって LC_1' は P' を再現したものである.

(3) **GC におけるアクションの非冗長性**

GC を構成しているアクション集合 B_k に関して

$$\cup\{B_k|k\in K\}\subseteq Act(P),\quad K=\{1,2,\cdots,m\}$$
$$B_i\cap B_j=\emptyset\ (i\neq j)$$

とする.

(4) **ゲートリスト G について**

ゲートリスト G は,

$$\{G\}=\cup\{B_k|k\in K\},\quad K=\{1,2,\cdots,m\}$$

とする. □

以上のような戦略を基に, モノリシックスタイルか

ら対応する制約指向スタイル1への変換アルゴリズムを構成する．

[アルゴリズム1]（モノリシックスタイルから制約指向スタイル1への変換）

入力：2.3節の［入力条件A］の2項目（$n=2$）

出力：2.3節の［定義2］に示された LC：LC_1, LC_2 と GC：GC_k ($k\in K$, $|K|\geq 1$)

ステップ1：GC を構成するアクション集合 $B_1, B_2, \cdots, B_m$ を［戦略1］に基づいて求める．

ステップ 1-1：変換されるプロセス P の展開過程において生じる **stop** を除いたすべてのプロセスに対して，そのイニシャルにプレアクションを要素として加えた集合を作る．ただし，プレアクションがないプロセスに関しては，イニシャルだけで1つの集合とする．これらの集合に対しては，作った順番に添字を付ける．

ステップ 1-2：ステップ 1-1 で作った集合のうち A_1 の要素だけで構成されている集合と，A_2 の要素だけで構成されている集合を取り除く．

ステップ 1-3：ステップ 1-2 で残った集合に対して，共通要素を持つ集合どうしの和集合をつくる．この和集合には2つの集合のうち添字の小さい方をつけ，2つの集合は取り除く．この操作を，すべての集合が共通要素を持たなくなるまで繰り返す．

ステップ 1-4：ステップ 1-3 で残った集合に対して，集合の添字の小さい順に集合名を $B_1, B_2, \cdots, B_m$ とする．

ステップ 2：$LC_1[A_1]$, $LC_2[A_2]$ と $GC_1[B_1]$, $GC_2[B_2], \cdots, GC_m[B_m]$ をそれぞれ求める．すなわち［定義12］の *Rest* を使って次のように求める．

$LC_1[A_1]\equiv Rest(P, A_1)$, $LC_2[A_2]\equiv Rest(P, A_2)$,
$GC_1[B_1]\equiv Rest(P, B_1), \cdots, GC_m[B_m]\equiv Rest(P, B_m)$ □

［アルゴリズム1］で作られた LC と GC は，次のような性質を持つ．

[性質1]（GC を構成するアクション集合に関する性質）

（1） $\cup\{Act(GC_k)\mid k\in K\}\subseteq Act(P)$, $K=\{1, 2, \cdots, m\}$

（2） $Act(GC_i)\cap Act(GC_j)=\emptyset$ $(i\neq j)$

（証明）［戦略1］の(3)に従っている［アルゴリズム1］と，*Rest* の定義より明らかである． □

[性質2]（アクションシーケンスに関する性質）変換されるプロセス P に対して，変換後のプロセスに存在しているアクションシーケンスは，次のような性質を持つ．

（1） 任意の GC_k に対し，任意の $s\in MTr(GC_k)$ はある $t\in MTr(P)$ の部分シーケンスである．

（2） 任意の GC_k に対し，ある $s\in MTr(GC_k)$ は A_1 と A_2 の交互シーケンスである．

（証明）［戦略1］の(2)-(b)および(3)に従っている［アルゴリズム1］と，*Rest* の定義より明らかである． □

[性質3]（連結プロセスと切断プロセスの再現に関する性質）変換されるプロセス P を展開する過程における任意のプロセス P' に対して，変換後のプロセスは，次のような性質を持つ．

（1） P' が A_1 と A_2 による連結プロセスである場合

この P' におけるものと同じ名前のプレアクションと同じイニシャルを持つプロセスが，LC_1 か LC_2 のいずれかの展開過程のプロセスに存在する．P' がプレアクションを持たない連結プロセスの場合は，同じイニシャルを持つプロセスが存在する．

（2） P' が A_1 と A_2 による切断プロセスである場合

この P' におけるものと同じ名前のプレアクションと同じイニシャルを持つプロセスが，ある GC_k の展開過程のプロセスに存在する．P' がプレアクションを持たない切断プロセスの場合は，同じイニシャルを持つプロセスが存在する．

（証明）［戦略1］の（2）に従っている［アルゴリズム1］と，*Rest* の定義より明らかである． □

[性質4]（連結プロセスと切断プロセスの再現数に関する性質）

（1） 変換されるプロセス P の展開過程における，A_1 と A_2 によるすべての連結プロセスが，LC の展開過程に同じ数だけ再現されており，GC には再現されていない．

（2） 変換されるプロセス P の展開過程における，A_1 と A_2 によるすべての切断プロセスが，GC の展開過程に同じ数だけ再現されており，LC には再現されていない．

（証明）［戦略1］の(1)から(3)の下で構成されている［アルゴリズム1］と，*Rest* の定義より明らかである． □

モノリシックスタイルのプロセスと［アルゴリズム1］によって得られるプロセス群に対して，次の定理が成立する．

[定理1] $P[A]$ をモノリシックスタイルのプロセスとし，$LC_1[A_1], LC_2[A_2]$ と $GC_1[B_1], BC_2[B_2], \cdots, GC_m[B_m]$ が［アルゴリズム1］で得られたとする．このとき P は，

$$Q \equiv (LC_1 | | | LC_2) | [G] | (GC_1 | | | GC_2 | | | \cdots | | | GC_m)$$
$$\{G\} = \cup \{B_k | k \in K\},\ K = \{1, 2, \cdots, m\}$$

に変換され，$P \sim Q$ が成立する．

（証明） 変換されるプロセス P と変換後のプロセス Q とが，強 bisimulation 等価であることを示すには，P と Q とが同じ LTS を生成することを示せば十分である．

そのため，あるプロセスに対して，そのプレアクションが生起しないとそのプロセスにはならないことを考慮し，以下の(A)と(B)を示せばよい．

(A) PとQとが同じイニシャルを持つことを示す．

(B) PとQを展開する過程における任意のプロセス P' と Q' に対して，P' と Q' のプレアクションが同じならば P' と Q' は同じイニシャルを持つことを示す．

ここで，Pを展開する過程で生じるあるプロセス P' は，［定義10］と［定義11］から，次の3種類に分けることができる．

(a)連結プロセス，(b)切断プロセス，(c)**stop**

(c)の **stop** は無動作プロセスであり，アクションの生起には関与しない．したがって(A)と(B)を議論する上で，(a)と(b)の場合について議論すればよい．

(A)の証明

Pは，［定義10］と［定義11］から，次の2つに分けることができる．

(a) プレアクションを持たない連結プロセス

(b) プレアクションを持たない切断プロセス

(a) Pがプレアクションを持たない連結プロセスの場合

Pにおいて生じ得る可能性があるアクションは，Pのイニシャルの要素のうちの1つである．［性質3］よりこの P のイニシャルと同じイニシャルを持つプロセスが，Q の LC_1 あるいは LC_2 のどちらかにだけ再現されている．そして，［性質4］より GC には再現されていない．

Pにおいて生じ得る可能性があるアクションは，そのアクションが生じた後のプロセスに関して，［定義10］と［定義11］から次の3つに場合分けできる．

①連結プロセスのプレアクションである場合

②切断プロセスのプレアクションである場合

③**stop** のプレアクションである場合

①の場合，［性質3］より Q において，このアクションは LC_1 あるいは LC_2 にだけ存在し，他のどのアクションとも同期をとらず単独で生起する．

②の場合，［性質3］より，Q において，このアクションは LC_1 あるいは LC_2 のどちらかとある1つの GC_k に存在し，これらが同期をとって生起する．

③の場合は，①の場合と同様である．

したがって，Q における LC_1 と LC_2 の独立並列合成，および $GC_1, GC_2, \cdots, GC_m$ の間の独立並列合成の関係，さらに両者の間の一般並列合成の関係により，P と Q は同じイニシャルを持つ．

(b) Pがプレアクションを持たない切断プロセスの場合

Pにおいて生じ得る可能性があるアクションは，Pのイニシャルの要素のうちの1つである．［性質3］よりこのPのイニシャルと同じイニシャルを持つプロセスが，Q のある GC_k にだけ再現されている．そして，［性質4］より LC には再現されていない．

Pにおいて生じ得る可能性があるアクションは，そのアクションが生じた後のプロセスに関して，(a)と同様に3つに場合分けできる．しかしすべての場合に対し，Q において，このアクションはある GC_k と LC_1 あるいは LC_2 にだけ存在し，これらが同期をとって生起する．

したがって，Q における LC_1 と LC_2 の独立並列合成，および $GC_1, GC_2, \cdots, GC_m$ の間の独立並列合成の関係，さらに両者の間の一般並列合成の関係により，P と Q は同じイニシャルを持つ．

(B)の証明

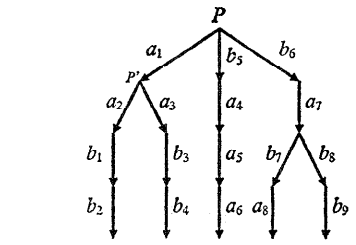
次に，P と Q を展開する過程で生じるあるプロセス P' と Q' が同じ名前のプレアクションを持つと仮定する．すると P' は，(a)連結プロセス，(b)切断プロセス，(c)**stop** のいずれかの場合に対応する．これらは(A)の場合と同様に，Q' における各プロセスの間の並列合成関係により，P' と Q' が同じイニシャルを持つことが容易に示せる．

以上より，P と Q は同じ LTS を生成し，［定理1］が成立する． □

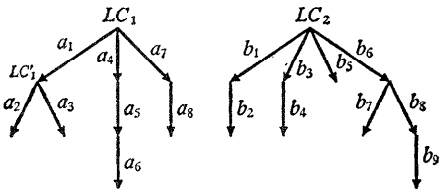
[例1]

次のようなプロセスPを考える（関連したデータ型の代数仕様については省略）．

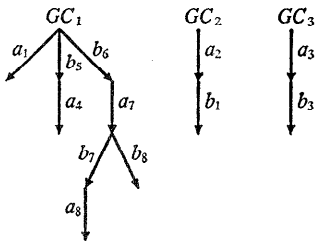
$$P \equiv a_1; (a_2; b_1; b_2; \mathbf{stop}[]a_3; b_3; b_4; \mathbf{stop})$$



(a) プロセスPの LTS
(a) LTS of Process P



(b) Local-constranit の LTS
(b) LTSs of the Local-constranit



(c) Grobal-constranit の LTS
(c) LTSs of the Global-constranit

図 3 例1の LTS
Fig. 3 LTSs of Example 1.

[]b_5; a_4; a_5; a_6; **stop**

[]b_6; a_7; (b_7; a_8; **stop**[]b_8; b_9; **stop**)

P に対応する LTS は図3(a)である．P と $Act(P)$ を分割した2つのアクション集合 $\{a_1, a_2, \cdots, a_8\}$ と $\{b_1, b_2, \cdots, b_9\}$ に対し，[アルゴリズム1]を適用すると，図3の(b)と(c)に示すような LTS で表現できる5つのプロセスを得る．図3(b)は LC に対応する LTS であり，図3(c)は GC に対応する LTS である．ここで，

$$Q \equiv (LC_1 |||LC_2) |[\Delta]| (GC_1 |||GC_2 |||GC_3)$$

とすると [定理1] により，Q は P と厳密に同じ LTS を生成する．ただし，$\{\Delta\} = \{a_1, a_2, a_3, a_4, a_7, a_8, b_1, b_3, b_5, b_6, b_7, b_8\}$ である． □

3.2 制約指向スタイル2への変換アルゴリズム

制約指向スタイル2は，制約指向スタイル1の特別な場合である．すなわち，LC と GC とを完全同期並列オペレータで結合したものである．この性質を満足させるために，[アルゴリズム1]で作られた LC と GC において，もし $\cup\{Act(GC_k) | k \in K\} \subsetneq Act(P)$ であれば，$Act(P)$ の要素でありかつ $\cup\{Act(GC_k) | k \in K\}$ の要素でないようなアクションを生じるプロセスを，GC に加える必要がある．したがって，これを次の戦略とする．

[戦略2] もし，$\cup\{Act(GC_k) | k \in K\} \subsetneq Act(P)$ であれば，$a \in Act(P)$ かつ $a \notin \cup\{Act(GC_k) | k \in K\}$ であるような各 a に対して，"a; **stop**" というプロセスを構成し，これを GC に加える．$\cup\{Act(GC_k) | k \in K\} = Act(P)$ であれば，何も加えない． □

以上の観点から，モノリシックスタイルから制約指向スタイル2へのアルゴリズムを，次のように構成する．

[アルゴリズム2] (モノリシックスタイルから制約指向スタイル2への変換)

入力：[アルゴリズム1]の入力と同じである．

出力：[アルゴリズム1]の出力と同じものに GC_l $(l = m+1, \cdots, m+n, n \geq 0)$ を加えたもの．

ステップ 1：[アルゴリズム1]のステップ1と同じ．

ステップ 2：[アルゴリズム1]のステップ2と同じ．

ステップ 3：もし，$\cup\{Act(GC_k) | k \in K\} = Act(P)$ であれば終了．

ステップ 4：ステップ2で求めた以外の GC：$GC_{m+1}[B_{m+1}], GC_{m+2}[B_{m+2}], \cdots, GC_{m+n}[B_{m+n}]$ をそれぞれ，[戦略2]に従って求める．ここで，

$$n = |Act(P) - \cup\{Act(GC_k) | k \in K}|,$$

$$B_{m+i} \cap B_{m+j} = \emptyset \ (i \neq j, 1 \leq i, j \leq n),$$

$$|B_{m+i}| = 1 \ (1 \leq i \leq n). \qquad \square$$

[アルゴリズム2]を適用して構成された各プロセスは，明らかに次のような性質を持つ．

[性質5] (GC を構成するアクション集合に関する性質)

$$\cup\{Act(GC_l) | l \in L\} = Act(P),$$

ただし，$L = \{1, 2, \cdots, m, m+1, m+2, \cdots, m+n\}$ □

[アルゴリズム2]では，GC_{m+1} から GC_{m+n} を，[定理1]で与えた Q の GC に，独立並列なプロセスとして加える．したがって，完全同期並列の意味と [定理1] および [性質5] を使って，次の定理が得られる．

[定理2] $P[A]$ をモノリシックスタイルのプロセスとし，$LC_1[A_1], LC_2[A_2]$ と $GC_1[B_1], GC_2[B_2], \cdots, GC_m[B_m], GC_{m+1}[B_{m+1}], \cdots, GC_{m+n}[B_{m+n}]$ が [アルゴ

リズム2] で得られたとする．このとき，P は，

$Q \equiv (LC_1 |||LC_2)$

$\|(GC_1 |||\cdots|||GC_m |||GC_{m+1} |||\cdots|||GC_{m+n})$

に変換され，$P \sim Q$ が成立する． □

4. 適 用 例

本論文で構成した記述スタイル変換アルゴリズムの有効性を示すために，"question/answer service[3]" に適用した例を示す．文献3)では，データ型を取り扱っているため，本論文ではデータを取り扱わないように改良した "改良型 question/answer service[8]" を使用する．

改良型 question/answer service では，**図4**に示すように，ユーザ Q によって質問 q_Q が生成され，サービスを通して別のユーザ A に転送される．ユーザ A はサービスから q_A を受けることで質問を確認し，これに対して応答 a_A を生成し，サービスを通してユーザ Q に転送する．ユーザ Q はサービスから a_Q を受けることで，応答を確認する．

このサービスを LOTOS のモノリシックスタイルで記述すると，以下のようになる（関連したデータ型の代数仕様については省略）．

process *QA_service* $[q_Q, q_A, a_Q, a_A]$: **noexit** :=
  q_Q ; q_A ; a_A ; a_Q ; **stop**
endproc

変換されるプロセスとして，この *QA_service* の動作式部分 $P \equiv q_Q ; q_A ; a_A ; a_Q$; **stop** を [アルゴリズム2] に与える．また，アクション集合の分割として，$A_1 = \{q_Q, a_Q\}$，$A_2 = \{q_A, a_A\}$ を与えると，[アルゴリズム2] は以下のような出力を与える（なお，この例では [アルゴリズム1] に与えても同じ結果を与える．）．

$LC_1 \equiv q_Q$; a_Q ; **stop**　$LC_2 \equiv q_A$; a_A ; **stop**

$GC_1 \equiv q_Q$; q_A ; **stop**　$GC_2 \equiv a_A$; a_Q ; **stop**

従って [定理2] により

$Q \equiv (q_Q$; a_Q ; **stop** $|||q_A$; a_A ; **stop**$)$
  $|||(q_Q$; q_A ; **stop** $|||a_A$; a_Q ; **stop**$)$

は，P と同じ LTS を生成し，強 bisimulation 等価な制約指向スタイルのプロセスである．

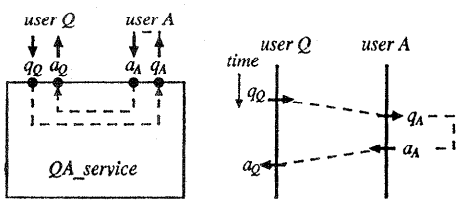


図4 改良型 question/answer service
Fig. 4 Modified question/answer service.

5. 考　　察

5.1 記述スタイル変換について

5.1.1 応用分野について

（1） 分割法としての能力

本論文の手法で分割されたプロセス（この場合 LC_1 と LC_2）どうしは，お互い完全に独立並列性を保つ．さらに，分割されたプロセスに対して，本手法が再適用できるため，n 個のプロセスに分割することが可能である．したがって，本手法は段階的詳細化に適した分割法である．

（2） 再構造化法としての能力

本手法を Expansion theorem[1),7)] とあわせて使用することで，構造化されていない LOTOS 仕様を構造化された制約指向スタイルに変換することが可能である．

（3） モジュール化法としての能力

本手法は，モノリシックスタイルで記述された1つのプロセスを意味のあるかたまりにモジュール化して分割することができる．また，LOTOS のモノリシックスタイルで記述された仕様はフラットであり，記述量が増えた場合，理解性が低下する．このような場合，本手法を適用して意味のあるパーティショニングを行うことで，理解性を向上させることが期待できる．

5.1.2 モノリシックスタイルから制約指向スタイルへの変換における利点

LOTOS の初心者が記述，理解しやすいモノリシックスタイルで仕様を記述し，新しい制約をつけ加えやすく，改良しやすい制約指向スタイルで仕様の改良作業を行うといったことが可能になる．

5.2 制限に対する考察

本論文で構成した LOTOS のモノリシックスタイルから制約指向スタイルへの変換法において，特に問題となる制限は，2.3 節で議論の簡単化のために導入した [制限B] の(2)，(3)，(4)である．

(2)と(3)については，本論文で構成した手法を基に，文献4）で用いられている手法のように各アクションに対して区別可能な添字をつけたり，また，区別可能なデータを送ることで，制限を取り除くように拡張することは容易である．

(4)については，再帰呼び出しを行う部分を切断プロセスの一種と考え，本論文で構成した手法を拡張す

れば，制限を取り除くことができる．

したがって，これらは本質的な制限ではない．

6. むすび

本論文では，LOTOS 仕様のモノリシックスタイルから制約指向スタイルへの等価性に基づく記述スタイル変換法を構成し，諸性質を導出するとともにその有用性を示した．

今後の課題として，変換アルゴリズム支援システムの構成などが挙げられる．

謝辞 熱心に討論して頂いた東北大学電気通信研究所高橋薫氏と程子学氏，ならびに東北大学白鳥研究室と野口研究室の諸氏に深く感謝します．

参考文献

1) ISO, Information Processing Systems—Open Systems Interconnection—LOTOS—A formal description technique based on the temporal ordering of observational behaviour, ISO 8807 (1989).
2) Turner, K. J.: Constraint-oriented style in LOTOS, *Proc. British Computer Society Workshop on Formal Methods in Standards* (1988).
3) Vissers, C. A., Scollo, G. and Sinderen, M. V.: Architecture and Specification Style in Formal Descriptions of Distributed Systems, *Protocol Specification, Testing, and Verification VIII*, North-Holland, pp. 189-204 (1988).
4) Langerak, R.: Decomposition of Functionality: a Correctness Preserving LOTOS Transformation, *Protocol Specification, Testing, and Verification X*, North-Holland, pp. 203-218 (1991).
5) Pavon, S., et al.: Inverse Expansion, *Proceedings of the FORTE '91 Fourth International Conference on: Formal Description Techniques*, pp. 303-318 (1991).
6) Eijk, P. V.: Tools for LOTOS Specification Style Transformation, *Formal Description Techniques II*, North-Holland, pp. 43-51 (1990).
7) Milner, R.: *Communication and Concurrency*, p. 260, Prentice-Hall (1989).
8) 郷健太郎，白鳥則郎：等価性に基づく LOTOS 仕様の系統的な分割アルゴリズム，電子情報通信学会情報ネットワーク研究会資料，IN 92-39 (1992).
9) Khendek, F., Bochmann, G. V. and Kant, C.: New Results on Deriving Protocol Specifications from Service Specifications, SIGCOM '89 Symposium Communications Architectures & Protocols, *Computer Communications Review*, Vol. 19, No. 4, pp. 136-145 (1989).
10) Higashino, T.: Service Specification and Its Protocol Specifications in LOTOS—A Survey for Synthesis and Execution—, *IEICE Trans. Fundamentals*, Vol. E 75-A, No. 3, pp. 330-338 (1992).
11) 馬渕博之，高橋 薫，白鳥則郎：LOTOS に基づいたプロトコル仕様の導出，電子情報通信学会情報ネットワーク研究会資料，IN 91-110 (1991).

(平成4年10月26日受付)
(平成5年4月8日採録)

郷 健太郎（正会員）

1968年生．1991年山梨大学工学部電気工学科卒業．1993年東北大学大学院工学研究科修士課程修了．現在同大学院情報科学研究科博士課程在学中．通信システムの仕様記述に関する研究に従事．

白鳥 則郎（正会員）

昭和21年生．昭和52年東北大学大学院博士課程修了．同年，東北大学電気通信研究所勤務．昭和59年，同大助教授（電気通信研究所）．平成2年，同大教授（工学部情報工学科）．情報通信システムの構成論，ソフトウェア開発法，ヒューマンインタフェースの研究に従事．情報処理学会25周年記念論文賞受賞．IEEE，電子情報通信学会，人工知能学会各会員．