6P−1 LOTOS仕様の等価性に基づく記述スタイル変換法とその応用*

郷 健太郎　白鳥 則郎†

東北大学工学部‡

1 はじめに

通信プロトコルや情報ネットワークのような大規模，複雑なシステムの設計には，一般に段階的詳細化が用いられる．仕様の段階的詳細化では，要求仕様を段階的に分割，詳細化を繰り返していくため，仕様の分割法が重要な役割を示す．LOTOS[1]は等価性に基づく数学的基盤を持つため，この段階的詳細化過程を等価性に基づいて行なえるという利点を持った形式記述技法(FDTs)である．LOTOSにおける段階的詳細化は変換である．前のステップと次のステップとで等価性を保つように仕様を変換して行くことで，要求仕様を満足する正しい実システムを構築することができる．

筆者らは既に文献[2]で，従来提案されていなかった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提案した．この変換法は，分割されたプロセスが(1)内部で同期をとらず，(2)すべて並列オペレータで結合されるという点で，最も分割能力が高いLOTOS仕様の分割法である．しかしながら文献[2]のアルゴリズムは(a)見通しが悪く，さらに(b)冗長なRemote-constraintのプロセスを作ってしまうことがある．そこで本稿では，この2点を改良したアルゴリズムを構成し，その適用例について述べる．

2 記述スタイル変換問題の形式化

モノリシックスタイルから制約指向スタイルへの変換問題は，以下のように示される．なお紙幅の都合上，記法や定義の詳細は省略する．詳細は文献[2]を参照されたい．

モノリシックスタイルから制約指向スタイルへの変換問題

以下の2つの入力が与えられたとする:

(1) モノリシックスタイルのプロセス B,

(2) アクション集合 $Act(B)$ を2分割した集合 A_1 と A_2．すなわち $A_1 \cup A_2 = Act(B)$ かつ $A_1 \cap A_2 = \emptyset$.

この時，以下の2つの条件を満足するLocal-constraint (LC)のプロセス LC_1, LC_2 とRemote-constraint(RC)のプロセス RC を見つける:

(a) $Act(LC_1) = A_1$, $Act(LC_2) = A_2$ であり，かつ $Act(RC) \subseteq Act(B)$.

(b) $(LC_1|||LC_2)|[G]|RC \cong B$, ここで $RC \equiv RC_1|||RC_2|||...|||RC_n$ $(n \geq 1)$, $G \subseteq Act(B)$ □

問題中で $\cong$ は観測合同である．

議論を簡単にするために，現在のバージョンでは以下のような制約を持ったプロセスを変換の対象とする:

(1) 内部アクションiを含まない，

(2) 再帰を持たない，

(3) 記述されているすべてのアクションが記述中に1回しか出現しない．

またLCのプロセス数は2とする．

3 LOTOS仕様の記述スタイル変換法

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[2] を用いて以下の戦略を構成する．

戦略 1

(1)LCの構成法

B 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

B を展開する過程における全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．

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

(3)RCにおけるアクションの非冗長性について

RCを構成しているアクション集合 C_k に関して

$\bigcup\{C_k \mid k \in \{1,2,...,n\}\} \subseteq Act(B)$

$C_i \cap C_j = \emptyset$ $(i \neq j)$

とする．

(4) 同期ゲート集合 G について

$G = \bigcup\{C_k \mid k \in \{1,2,...,n\}\}$

とする． □

以上の戦略をもとに，2節で述べた記述スタイル変換問題を解決するアルゴリズムを構成する．

アルゴリズム 1 (LOTOS仕様の変換)

入力:(1) 変換されるプロセス B と (2) アクション集合 A_1, A_2

出力: LC_1, LC_2 と RC $(RC_1|||... |||RC_n, n \geq 1)$

ステップ1:RCを構成するアクション集合 $C_1, C_2, ..., C_n$ を戦略1に基づいて求める．

---

*A LOTOS Specification Style Transformation Method and Its Application to OSI services

†Kentaro GOH and Norio SHIRATORI

‡Faculty of Engineering, Tohoku University

ステップ 1.1: 変換されるプロセス B 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，全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 $C_1, C_2, ..., C_n$ とする．

ステップ 2: LC_1, LC_2 と $RC_1, RC_2, ...RC_n$ をそれぞれ求める．すなわち写像 Rest を使って次のように求める．

$LC_1 \equiv Rest(B, A_1)$, $LC_2 \equiv Rest(B, A_2)$

$RC_1 \equiv Rest(B, C_1)$, ..., $RC_n \equiv Rest(B, C_n)$ □

モノリシックスタイルのプロセス B とアルゴリズム 1 によって得られたプロセスに対して，次の定理が成立する．

定理 1

B をモノリシックスタイルのプロセスとし，LC_1, LC_2 と $RC_1, RC_2, ..., RC_n$ がアルゴリズム 1 で得られたとする．このとき B は，

$D \equiv (LC_1 ||| LC_2) |[G]| (RC_1 ||| RC_2 ||| ... ||| RC_n)$

$G = \bigcup \{C_k \mid k \in \{1, 2, ..., n\}\}$

に変換され，$B \cong D$ が成立する． □

証明は，B 中の任意のプロセス B' と D 中の任意のプロセス D' に対して，プレアクションが同じであれば同じイニシャルを持つことを示すことによって行なう．ここでは紙幅の都合上省略する．

4 適用例

アルゴリズム 1 を Abracadabra service[3] に適用する例を示す．Abracadabra service を LOTOS のモノリシックスタイルで記述すると次のようになる:

$B \equiv ConReq; ConInd; ConRes; ConCnf;$

$DatReq; DatInd; DisReq; DisInd;$ **stop**

今アクション集合を，(1) Sender 側のアクション A_1 と (2) Receiver 側のアクション A_2 とに分けるとする．

$A_1 = \{ConReq, ConCnf, DatReq, DisReq\}$

$A_2 = \{ConInd, ConRes, DatInd, DisInd\}$

以上の B と A_1, A_2 をアルゴリズムに入力すると，以下のステップを経て B が制約指向のプロセスに変換される．

アルゴリズムの適用過程

ステップ 1: ここでは，RC を構成するアクション集合を得るために，以下の 4 ステップの操作を行う:

ステップ 1.1: ここでは，以下の集合が得られる:

$\{ConReq\}_1$, $\{ConReq, ConInd\}_2$,

$\{ConInd, ConRes\}_3$, $\{ConRes, ConCnf\}_4$,

$\{ConCnf, DatReq\}_5$, $\{DatReq, DatInd\}_6$,

$\{DatInd, DisReq\}_7$, $\{DisReq, DisInd\}_8$

ステップ 1.2: 1, 3, 5 の添字を持つ集合が除かれ，以下の集合だけが残る:

$\{ConReq, ConInd\}_2$, $\{ConRes, ConCnf\}_4$,

$\{DatReq, DatInd\}_6$, $\{DatInd, DisReq\}_7$,

$\{DisReq, DisInd\}_8$

ステップ 1.3: 添字 6 と 7 の集合で $DatInd$ が，添字 7 と 8 の集合で $DisReq$ が共通しているため，これら 3 つの集合の和集合をつくり，添字を 6 とする:

$\{ConReq, ConInd\}_2$, $\{ConRes, ConCnf\}_4$,

$\{DatReq, DatInd, DisReq, DisInd\}_6$

ステップ 1.4: 以下のように集合名がつけられる:

$\{ConReq, ConInd\}_2 \equiv C_1$,

$\{ConRes, ConCnf\}_4 \equiv C_2$,

$\{DatReq, DatInd, DisReq, DisInd\}_6 \equiv C_3$

ステップ 2: Rest を使って以下のプロセスが得られる:

$LC_1 \equiv ConReq; ConCnf; DatReq; DisReq;$ **stop**

$LC_2 \equiv ConInd; ConRes; DatInd; DisInd;$ **stop**

$RC_1 \equiv ConReq; ConInd;$ **stop**

$RC_2 \equiv ConRes; ConCnf;$ **stop**

$RC_3 \equiv DatReq; DatInd; DisReq; DisInd;$ **stop**

ここで定理 1 により，$(LC_1 ||| LC_2) ||| (RC_1 ||| RC_2 ||| RC_3)$ と B とは観測合同であり，これはモノリシックスタイルの B を制約指向スタイルに変換したものに相当する．

5 おわりに

本稿では，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ついて述べた．現在，この変換支援環境を WS 上に実装中である．今後の課題として，支援環境を実際に使用して評価すること，変換可能な仕様の範囲を広げることなどが挙げられる．

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] 郷 健太郎, 白鳥 則郎 : "LOTOS 仕様の系統的な分割アルゴリズム", 信学技報, IN92-39 (1992).

[3] ISO, *Information technology — Open Systems Interconnection — Guidelines for the application of Estelle, LOTOS and SDL*, ISO/IEC TR 10167 (1991).