ラベル付き遷移システムで記述された要求仕様の並列プロセス群への一分解法

喜家村 奨[†] 高田喜朗[†] 関 浩之[†]

並列処理システムの要求仕様からそれを実現する並列プロセス群からなる実装レベル仕様への分解法を提案する．仕様は，プロセスと環境間のインタフェース情報と，各成分プロセスの動作定義からなると考えられる．インタフェース情報とは，成分プロセスの総数および各成分プロセスごとにその入出力イベントの集合（アルファベット）を定めたものである．一方，プロセスの動作定義は遷移システム（LTS）で与えるものとする．本論文では，要求仕様，および実装レベル仕様におけるインタフェース情報が与えられたとき，各実装プロセスの動作定義を与える LTS を自動生成する方法を提案する．また，要求仕様と生成された実装レベル仕様の動作の等価性を証明する．さらに，実装レベル仕様において，各成分プロセスの状態数およびプロセス間の同期のためのメッセージ数を減らす手法についても考察する．

A Method of Decomposing a Labeled Transition System into Parallel Processes

Susumu Kiyamura,[†] Yoshiaki Takata[†] and Hiroyuki Seki[†]

A method of generating an implementation level specification (abberviated as an implementation) of a parallel system is proposed. We consider a specification consists of an interface condition and a behavior definition. For each component process, an interface condition specifies the set of events which the process should participate in. A behavior definition of a component process is given by a labeled transition system (LTS). In this paper, for a given requirements specification and a given interface condition of an implementation, a method of automatically generating a behavior definition (LTS) of an implementation is proposed. The behavioral equivalence (weak bisimulation) between a requirements specification and the generated implementation is also proved. Furthermore, a method of decreasing the number of process states as well as decreasing the number of internal messages for synchronizing component processes is presented.

1. まえがき

大規模な並列システムを構築する場合，要求仕様の決定から実装までを多数のフェーズにわけて設計する．しかし，従来の方法では，各フェーズの仕様作成は人手で行われており，仕様作成やフェーズ間の整合性の検証に多くの時間が必要となっている．

本論文では，単一プロセスからなる要求仕様から複数のプロセスからなる実装レベル仕様への分解法について考察する．並列プロセスの仕様は，プロセスと環境間の**インタフェース情報**と，各成分プロセスの**動作定義**からなると考えられる．インタフェース情報とは，成分プロセスの総数および各成分プロセスごとにその入出力イベントの集合（アルファベット）を定めたものである．一方，プロセスの動作定義は遷移システム（LTS）で与えるものとする．さて，利用可能プロセス数や入出力ポートの配置等の実装上の制約や対コスト性能比の要求等から，実装レベル仕様におけるインタフェース情報はあらかじめ決定されている場合が多い．そこで，本論文では，要求仕様，および実装レベル仕様における環境とのインタフェース情報が与えられたとき，各実装プロセスの動作定義を与える LTS を求める問題を LTS 分割問題として定義し， LTS 分割問題を解くアルゴリズムを提案する．

通信プロトコルの分野では，サービス定義からプロトコル仕様を自動合成する研究が多数行われている[8]．要求仕様として LTS を用いた従来の研究のうち，要求仕様と実装レベル仕様の等価性を証明している代表的なものと本手法の特長を**表 1** にまとめる．文献 6)

† 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

表 1 LTS を要求仕様とする分割法の比較
Table 1 Comparison of decomposition methods.

	アルファベット	分割数	同期の手段	冗長な同期の削除	等価性の証明
Langerak[6]	互いに素	2	内部イベント	部分的にあり	弱双模倣等価
馬渕ら[7]	互いに素	≥ 2	内部イベント	なし	弱双模倣等価
郷ら[1]	互いに素	2	外部イベント	あり	強双模倣等価
本研究	制約なし	≥ 2	内部イベント	あり	弱双模倣等価

の方法は，表 1 に示した前提条件が成り立つときのみ，本論文ステップ 3（後述）に相当する最適化を行っているが，最適化を行った場合の厳密な等価性の証明は与えられていない．文献 7) の方法は本手法のステップ 1（後述）に相当する．文献 1) の方法は 2 つのプロセスの同期を制御するプロセス群を配置する点で文献 6)，7) や本手法とアプローチが異なる．有限状態モデルより広いモデルを用いたものとして，たとえば文献 2) では，レジスタを持つ拡張 FSM（EFSM）を用い，1 つの EFSM で表された要求仕様からプロセス間で交換されるメッセージ数が最小となる実装レベル仕様を，0-1 整数線形計画法を用いて合成する方法が示されている．文献 4) では時間制約付き FSM で定義された要求仕様を扱っている．ただし，これら拡張モデルを用いた研究でも，本論文で行うような冗長な同期の削除は考慮されていない．

次に，本論文の構成を示す．2 章では本論文で前提とする並列プロセスモデルを定義する．3 章では提案する変換アルゴリズムを詳しく述べる．また，変換前の LTS と，変換によって得られた LTS との弱双模倣等価性[9] を証明する（定理 1）．4 章では各成分プロセスの状態数や同期イベントの数を削減するための改良点について説明する．特に，状態数削減の手法として，インタリーブ領域（各プロセスが互いに素なイベントを非同期で実行するような領域）に着目した．5 章では，他の文献と同じ仮定をした場合について考察し，この場合も提案アルゴリズムが適用可能であることを述べる．

2. 準　　備

2.1 要求仕様と実装レベル仕様との関係

本論文で扱う問題の概要を（**図 1**）に示す．まず，要求仕様（図 1(a)），および実装レベル仕様におけるインタフェース情報（図 1(b)）が与えられる．前者は単一プロセス（$L_{\rm in}$ と呼ぶ）のインタフェース情報

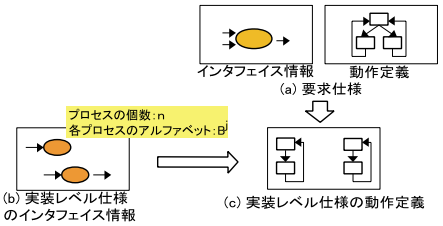

図 1 要求仕様と実装レベル仕様の関係
Fig. 1 Relation between a requirements specification and an implementation level specification.

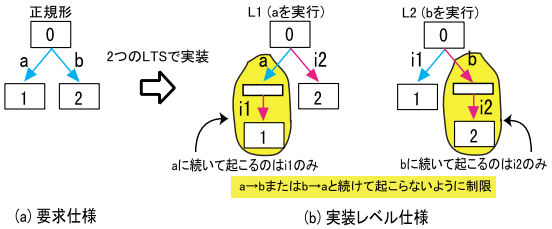

図 2 実装レベル仕様におけるプロセス間の同期
Fig. 2 Synchronization in an implementation level specification.

（アルファベット）および動作定義からなる．後者は，実装レベル仕様のプロセスの個数 (n) と各プロセス $L^j_{\rm out}$ $(1 \leq j \leq n)$ のアルファベット B^j からなる．この両者から実装レベル仕様の各プロセスの動作定義（図 1(c)）を導出するのがここでの問題である．

本論文ではプロセスの動作定義を連続イベント制約付き遷移システムで与える．これはイベントの発生順に制限を持たせた LTS である．以降，特に断らない限り，連続イベント制約付き遷移システムのことを単に LTS と呼ぶ．たとえば**図 2**(a) の要求仕様を 2 つのプロセス（$L1$ と $L2$）で実装し，イベント a は $L1$ のみ，イベント b は $L2$ のみが実行する場合を考える．このとき，各プロセスは相手がイベントを実行したかどうかを知る必要がある．そこで各 LTS がイベントを実行した後，それを他方の LTS に通知する同期イベントを発生させる（図 2(b)）．この同期イベントの発生の前に他のイベントが発生しないようにシステム全体に制約をつけなければならない（たとえばイベント a が発生した後はそれを通知する同期イベント $i1$ が発生しない限り他のイベントは発生できないようにする）．このようなイベントの発生順に関する制約を連続イベント制約と呼ぶ．連続イベント制約は $(0, a, 0', i1)$ のように状態，外部イベント，中間状態，同期イベントの組で記述する．なお，文献 2)，4) 等で行っているように，「$L_{\rm in}$ で図 2(a) のような遷移があるときは，a，b は同じプロセスに割り当てられる

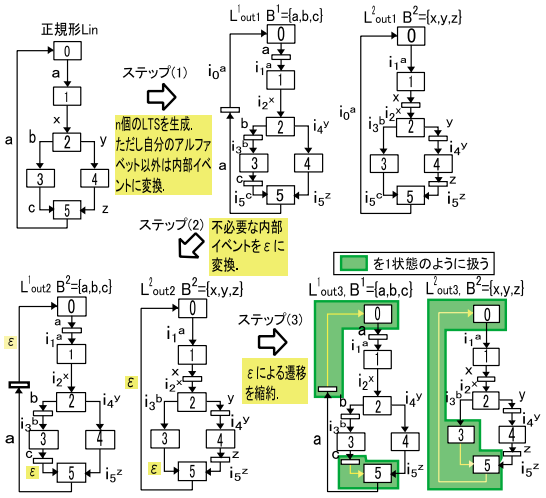

図 3 変換ステップの概略
Fig. 3 Outline of the transformation steps.

(*1)」と仮定するならば，連続イベント制約は不要となる．すなわち，連続イベント制約は上記の仮定より強くない制約である．(*1) を仮定し，代わりに連続イベント制約を用いない場合でも，本手法はそのまま適用できる．これについては 5 章で述べる．L_{in} から実装レベル仕様の各 LTS への変換は次の 3 つのステップからなる（**図 3**）．

ステップ (1) L_{in} を n 個コピーし，$L^1_{\mathrm{out1}}, \ldots, L^n_{\mathrm{out1}}$ とする．ここで，

- LTS L^j_{out1} がイベント a に関与するとき（$a \in B^j$ のとき），イベント a に関与しないプロセスと同期をとるためのイベント i を a の直後に挿入する．a と i は連続イベント制約を満たすとする．
- LTS L^j_{out1} がイベント a に関与しないとき（$a \notin B^j$ のとき），a を上記の同期イベント i に置き換える．

ステップ (2) L^j_{out1} において不必要な同期イベントを ϵ に変更する（得られた各 LTS を L^j_{out2} とする）．

ステップ (3) L^j_{out2} において ϵ による遷移を縮約する（得られた各 LTS を L^j_{out3} とする）．

ステップ 3 で得られた各 LTS L^j_{out3} を成分とする並列プロセスが，実装レベル仕様となる．

2.2 LTS の定義

組 $s \in S^1 \times \cdots \times S^k$ の第 j 成分を $s^{[j]}$ と書く．$S^1 = \cdots = S^k = S$ のとき，$S^1 \times \cdots \times S^k$ を $(S)^k$ と書く．

定義 1（イベント集合） イベントすべての集合を Act とする．Act は外部から観測可能なイベントと観測可能ではない内部イベント τ からなる集合である．

定義 2（LTS） 連続イベント制約付き遷移システム（LTS）は以下の 5 字組．

$(A, S, \rightarrow, s_0, C)$
where
$A \subseteq Act$：イベントの集合（アルファベット），
S： 状態の集合，
$\rightarrow \subseteq S \times A \times S$： 遷移関係，
$s_0 \in S$： 初期状態，
$C \subseteq S \times (A - \{\tau\}) \times S \times (A - \{\tau\})$：
　連続イベント制約．

$(s, \alpha, s') \in \rightarrow$ であることを $s \xrightarrow{\alpha} s'$ と書く．$s \xrightarrow{\alpha} s'$ であるような s' が存在することを $s \xrightarrow{\alpha} \cdot$ と書き，そのような s' が存在しないことを $s \not\xrightarrow{\alpha} \cdot$ と書く．
$\rightarrow$ は以下の性質を満たさなければならない．

$$(s, a, s', b) \in C \wedge s \xrightarrow{a} s' \supset s' \not\xrightarrow{\alpha} \cdot \text{ for } \forall \alpha \in A - \{b\}. \quad (1)$$

□

条件 (1) は，連続イベント制約 $(s, a, s', b) \in C$ があるとき，イベント a を実行した直後の状態においては，b 以外のイベント（τ を含む）による遷移が定義されるのを禁止している．

2 つの LTS $L^0 = (A, S, \rightarrow, s_0, \emptyset)$，$L^1 = (A, S, \rightarrow, s_0, C)$ について，L^0 と L^1 の動作はまったく同じであるので，LTS を個別に扱う場合は連続イベント制約に本質的意義はない．しかし，以下に述べる直積 LTS を考えた場合，成分 LTS の連続イベント制約は直積 LTS の動作に影響する．

複数の成分プロセスからなる並列プロセスの動作は，以下の直積 LTS で定義される．

定義 3（直積 LTS） k 個の LTS，$L^j = (A^j, S^j, \rightarrow_j, s^j_0, C^j)$ $(j \in [1, k])$ を成分とする直積 LTS を $L^1 \times \cdots \times L^k$ と書き，以下のように定義する．

$L^1 \times \cdots \times L^k := (A, S, \rightarrow, s_0, C)$
where
$A := \bigcup_{j \in [1,k]} A^j$,
$S := S^1 \times \cdots \times S^k$,
$s \xrightarrow{\alpha} s' :\Leftrightarrow$
もし $\alpha \neq \tau$ ならば，
任意の $j \in [1, k]$ について，$\alpha \in A^j$ ならば $s^{[j]} \xrightarrow{\alpha}_j s'^{[j]}$，$\alpha \notin A^j$ ならば $s^{[j]} = s'^{[j]}$.
もし $\alpha = \tau$ ならば，
ある $j \in [1, k]$ について $s^{[j]} \xrightarrow{\tau}_j s'^{[j]} \wedge s^{[j']} = s'^{[j']}$ for $\forall j' \neq j$.
加えて，いずれの場合も s，α が以下の条件を満たす．

• ある $j \in [1,k]$, $t \in S^j$, $a,b \in A^j$ が存在し, $t \xrightarrow{a}_j s^{[j]}$, $(t,a,s^{[j]},b) \in C^j$ ならば $\alpha = b$ (イベント a で状態 $s^{[j]}$ に至ったならば, $s^{[j]}$ からの遷移は必ずイベント b によって起こる).

$s_0 := (s_0^1, \ldots, s_0^k)$,

$(s,a,s',b) \in C \ :\Leftrightarrow$

$(s^{[j]}, a, s'^{[j]}, b) \in C^j$ for $\exists j \in [1,k]$.

□

→ の定義において, 10〜13 行目は次のことを表している. ある LTS L^j が連続イベント制約 $(t,a,s^{[j]},b)$ を持つとき, たとえ L^j 以外の LTS L^i において $s^{[i]} \xrightarrow{\alpha}_i \cdot$ $(\alpha \neq b)$ だったとしても, 連続イベント制約が優先し, $L^1 \times \cdots \times L^k$ においては $s \not\xrightarrow{a} \cdot$ となる. 1〜3 行目は (7〜9 行目の条件が成り立つとき) L^j が α をアルファベットに含んでいれば $s'^{[j]}$ に遷移し, 含んでいなければ状態は変化しないことを表す.

定義 4 (イベントの隠蔽) LTS $L = (A,S,\rightarrow,s_0,C)$ とイベント集合 $H \subseteq Act - \{\tau\}$ に対し, LTS $L\backslash H$ を以下のように定義する.

$L\backslash H := (A \cup \{\tau\} - H, S, \rightarrow_{\mathrm{h}}, s_0, C_{\mathrm{h}})$

where

$s \xrightarrow{\alpha}_{\mathrm{h}} s' \ :\Leftrightarrow \ s \xrightarrow{\alpha} s' \ \wedge \ \alpha \notin H$, or

$s \xrightarrow{\beta} s'$ for $\exists \beta \in H \ \wedge \ \alpha = \tau$.

$(s,a,s',b) \in C_{\mathrm{h}} \ :\Leftrightarrow$

$(s,a,s',b) \in C \wedge a \notin H \wedge b \notin H$.

□

定義 5 Act のある 1 つの要素を ϵ と呼び, 他と区別する. 特に断わらない限り, 各 LTS は ϵ をアルファベットに持たないとする. 任意のイベント $\alpha \in Act$ について $\hat{\alpha}$ を以下のように定義する.

$$\hat{\alpha} := \begin{cases} \alpha & \text{if } \alpha \neq \tau, \\ \epsilon & \text{if } \alpha = \tau. \end{cases}$$

□

定義 6 (観測可能な遷移関係) LTS $L = (A,S,\rightarrow,s_0,C)$ に対し, 関係 $\Rightarrow_L \ \subseteq S \times Act \times S$ を, 以下を満たす最小の関係と定義する. ただし, $(s,\alpha,s') \in \Rightarrow_L$ であることを $s \overset{\alpha}{\Rightarrow}_L s'$ と書く.

$s \overset{\epsilon}{\Rightarrow}_L s$,

$s \overset{\epsilon}{\Rightarrow}_L s' \ \wedge \ s' \xrightarrow{\tau} t \ \supset \ s \overset{\epsilon}{\Rightarrow}_L t$,

$s \overset{\epsilon}{\Rightarrow}_L s' \ \wedge \ s' \xrightarrow{\alpha} t' \ \wedge \ t' \overset{\epsilon}{\Rightarrow}_L t \ \supset \ s \overset{\alpha}{\Rightarrow}_L t$.

□

2.3 双模倣等価

定義 7 (弱双模倣関係) $L^1 = (A^1,S^1,\rightarrow_1,s_0^1,C^1)$, $L^2 = (A^2,S^2,\rightarrow_2,s_0^2,C^2)$ を任意の LTS とする. 関係 $\mathcal{R} \subseteq S^1 \times S^2$ が組 (L^1,L^2) に対する弱双模倣関係であるとは, $\forall (s_1,s_2) \in \mathcal{R}$ と $\forall \alpha \in Act$ について以下の 2 条件が成り立つことである.

(a) $\exists s_1' \in S_1$ に対して $s_1 \xrightarrow{\alpha}_1 s_1'$ ならば, $\exists s_2' \in S_2$ が存在し, $s_2 \overset{\hat{\alpha}}{\Rightarrow}_2 s_2'$ かつ $(s_1',s_2') \in \mathcal{R}$,

(b) $\exists s_2' \in S_2$ に対して $s_2 \xrightarrow{\alpha}_2 s_2'$ ならば, $\exists s_1' \in S_1$ が存在し, $s_1 \overset{\hat{\alpha}}{\Rightarrow}_1 s_1'$ かつ $(s_1',s_2') \in \mathcal{R}$ □

定義 8 (弱双模倣等価) 任意の LTS $L^1 = (A^1,S^1,\rightarrow_1,s_0^1,C^1)$, $L^2 = (A^2,S^2,\rightarrow_2,s_0^2,C^2)$ について, $(s_0^1,s_0^2) \in \mathcal{R}$ であるような (L^1,L^2) に対する弱双模倣関係 $\mathcal{R}$ が存在するとき, L^1 と L^2 は弱双模倣等価であるといい, $L^1 \approx L^2$ と書く. □

3. LTS 分割問題を解くアルゴリズム

この章では, 本論文で扱う問題である LTS 分割問題を定義し, これを解くアルゴリズムを述べる.

定義 9 (LTS 分割問題) 以下の入出力で定義される問題を LTS 分割問題という.

入力: $L_{\mathrm{in}} = (A,S,\rightarrow_{\mathrm{in}},s_0,\emptyset)$: LTS.

n: 自然数.

$B^j \subseteq A$ $(j \in [1,n])$. ただし $B^1 \cup \cdots \cup B^n = A - \{\tau\}$.

出力: $L_{\mathrm{out}}^j = (A^j,S^j,\rightarrow_j,s_0^j,C^j)$ for $\forall j \in [1,n]$: LTS の系列.

ただし $B^j \subseteq A^j, (A^j - B^j) \cap (A \cup \{\tau\}) = \emptyset$ かつ, $Act_{\mathrm{sync}} = \bigcup_{j=1}^n (A^j - B^j)$ とおくとき, $L_{\mathrm{in}} \approx (L_{\mathrm{out}}^1 \times \cdots \times L_{\mathrm{out}}^n) \backslash Act_{\mathrm{sync}}$.

Act_{sync} は実装レベル仕様内で用いられる同期のためのイベント集合である.

□

以降, 定義 9 の入力 L_{in}, A, S, $\rightarrow_{\mathrm{in}}$, s_0, B^j および n を固定する. ここで L_{in} は以下を満たすと仮定する.

仮定 10 任意の $s,s_1,s_2 \in S$, $a \in A - \{\tau\}$ について, $s \xrightarrow{a}_{\mathrm{in}} s_1 \wedge s \xrightarrow{a}_{\mathrm{in}} s_2 \wedge s_1 \neq s_2$ ならば, $a \in B^j$ となる $j \in [1,n]$ はちょうど 1 つ. □

もし L_{in} が仮定 10 を満たさない場合, そのようなすべての $s,s_1,s_2 \in S$, $a \in A - \{\tau\}$ について, 遷移 $s \xrightarrow{a}_{\mathrm{in}} s_2$ を削除した後, 新しい状態 s_2' および遷移 $s \xrightarrow{\tau}_{\mathrm{in}} s_2'$, $s_2' \xrightarrow{\tau}_{\mathrm{in}} s$, $s_2' \xrightarrow{a}_{\mathrm{in}} s_2$ を追加することで, 仮定 10 を満たす LTS を得ることができる. かつ, この LTS は元の L_{in} と弱双模倣等価である. よって仮定 10 を置いても一般性は失われない.

提案アルゴリズムが生成する実装レベル仕様および中間的な仕様においては，S，A，B^j に加えて以下の状態およびイベントの集合が用いられる．

定義 11 $L_{\rm in} = (A, S, \rightarrow_{\rm in}, s_0, \emptyset)$ に対し，集合 $S_{\rm aux}$ および $A_{\rm aux}$ を以下のように定義する．

$$S_{\rm aux} := S \times (A - \{\tau\}) \times S,$$
$$A_{\rm aux} := \{(a, s') \in A \times S \mid \exists s \in S,\ s \xrightarrow{a}_{\rm in} s'\}.$$

$S_{\rm aux}$ の要素は状態として，$A_{\rm aux}$ の要素は同期イベントとして用いられる．見やすさのため，$S_{\rm aux}$ の要素 (s, a, s') を $t^a_{s,s'}$，$A_{\rm aux}$ の要素 (a, s) を i^a_s と書く．$A_{\rm aux}$ の部分集合 A_τ を以下のように定義する．

$$A_\tau := A_{\rm aux} \cap (\{\tau\} \times S).$$

□

以下では，2.1 節の各ステップについて詳細を述べる．

3.1 ステップ (1)：$L^j_{\rm out1}$ を求める

ステップ (1) では，まず，要求仕様を表す LTS（$L_{\rm in}$）を n 個複製する．次に複製した各 LTS について，そのアルファベットに含まれないイベントを同期イベントに変換する．アルファベットに含まれるイベントについては，そのイベントの実行後，その実行を他の LTS に知らせる同期イベントを挿入する．以下に $L^j_{\rm out1}$ の定義を示す．

定義 12 $L_{\rm in}$，B^j $(j \in [1, n])$ に対して $L^j_{\rm out1}$ を以下のとおり定義する．

$$L^j_{\rm out1} = (A^j_1, S \cup S_{\rm aux}, \rightarrow_{1,j}, s_0, C^j_1)$$
where
$$A^j_1 = B^j \cup A_{\rm aux},$$
$\rightarrow_{1,j}, C^j_1$：以下を満たす最小の関係：
for $\forall s, s' \in S, a \in A$,
$$a \in B^j \ \wedge\ s \xrightarrow{a}_{\rm in} s' \ \supset$$
$$s \xrightarrow{a}_{1,j} t^a_{s,s'} \ \wedge\ t^a_{s,s'} \xrightarrow{i^a_{s'}}_{1,j} s' \ \wedge$$
$$(s, a, t^a_{s,s'}, i^a_{s'}) \in C^j_1,$$
$$a \notin B^j \ \wedge\ s \xrightarrow{a}_{\rm in} s' \ \supset\ s \xrightarrow{i^a_{s'}}_{1,j} s'.$$

□

3.2 ステップ (2)：$L^j_{\rm out2}$ を求める

ステップ (2) では，ステップ (1) で得られた $L^j_{\rm out1}$ の中で，実装レベル仕様においては必要でない同期イベントを ϵ に変換する．実装レベル仕様において必要な同期イベントとは，以下で述べる実行可能通知イベントおよび実行不能通知イベントである．

実行不能通知イベントとは，$L_{\rm in}$ のある状態 s から 2 つのイベント a, b による遷移があり，一方のみ（仮に b とする）をアルファベットに持つ成分 LTS（**図 4**(a) の $L^1_{\rm out1}$）が存在するときに，他方（ここでは a）を実行した LTS ($L^2_{\rm out1}$) が実行する同期イベント（図 4(a) の $i^a_{s_1}$）のことである．すなわちこの同期イベントは，b の実行が禁止されたことを他の LTS に通知している．

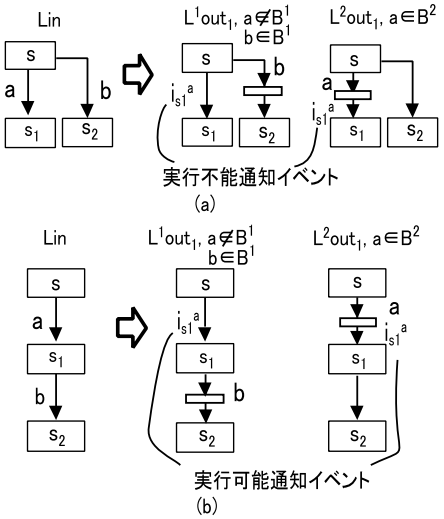

図 4 実行不能，可能通知イベント
Fig. 4 Disabling and enabling events.

実行可能通知イベントとは，$L_{\rm in}$ のある状態 s から s_1 への a による遷移があり，s_1 から s_2 への b による遷移があるとき，a をアルファベットに含まないが，b は含むような LTS（図 4(b) の $L^1_{\rm out1}$）に対して，a を実行し状態 s_1 に到達した LTS($L^2_{\rm out1}$) が，b の実行が可能になったことを通知するイベント（図 4(b) の $i^a_{s_1}$）のことである．

$L_{\rm in}$ が内部イベント τ による遷移を含む場合は，以上の実行可能（不能）通知イベントの定義を，τ による遷移を 0 回以上行った後に行える遷移に関して拡張する必要がある．準備として τ 閉包を以下のように定義する．

定義 13 $L_{\rm in}$ の任意の状態 $s \in S$ に対し，以下の条件を満たす最小の集合を $CL_\tau(s)$ と書く．

$$s \in CL_\tau(s),$$
$$s' \in CL_\tau(s) \ \wedge\ (s' \xrightarrow{\tau}_{\rm in} s'') \ \supset\ s'' \in CL_\tau(s).$$

□

実行不能通知イベントおよび実行可能通知イベントの形式的定義は以下のようになる．

定義 14 $L_{\rm in}$ および $B^1, \ldots, B^n$ において，

$$s \xrightarrow{a} s_1 \ \wedge\ s' \in CL_\tau(s) \ \wedge\ s' \xrightarrow{b} s_2 \ \wedge$$
$$a \notin B^j \ \wedge\ b \in B^j$$

が成り立つような $a,b \in A$, $s,s_1,s_2 \in S$ および $j \in [1,n]$ が存在するとき，$i^a_{s_1}$ を L^j_{out1} における実行不能通知イベントと呼ぶ．同時に，$a \in B^k$ である任意の $k \in [1,n]$ について，$i^a_{s_1}$ を L^k_{out1} における実行不能通知イベントと呼ぶ．
任意の $j \in [1,n]$ について，L^j_{out1} における実行不能通知イベントすべてからなる集合を $Disabling^j$ とする．
□

定義 15 L_{in} および $B^1,\ldots,B^n$ において，

$$s \xrightarrow{a} s_1 \ \wedge\ s'_1 \in CL_\tau(s_1) \ \wedge\ s'_1 \xrightarrow{b} s_2 \ \wedge\ a \notin B^j \ \wedge\ b \in B^j$$

が成り立つような $a,b \in A$, $s,s_1,s_2 \in S$ および $j \in [1,n]$ が存在するとき，$i^a_{s_1}$ を L^j_{out1} における実行可能通知イベントと呼ぶ．同時に，$a \in B^k$ である任意の $k \in [1,n]$ について，$i^a_{s_1}$ を L^k_{out1} における実行可能通知イベントと呼ぶ．
任意の $j \in [1,n]$ について，L^j_{out1} における実行可能通知イベントすべてからなる集合を $Enabling^j$ とする．
□

以下に L^j_{out2} の定義を示す．

定義 16 L^j_{out1} $(j \in [1,n])$ に対し，L^j_{out2} を以下のとおり定義する．

$$L^j_{\mathrm{out2}} = (A^j_2 \cup \{\epsilon\}, S \cup S_{\mathrm{aux}}, \rightarrow_{2,j}, s_0, C^j_2)$$

where

$A^j_2 = B^j \cup Disabling^j \cup Enabling^j$,

$\rightarrow_{2,j}$: 以下を満たす最小の関係：

for $\forall s,s' \in S \cup S_{\mathrm{aux}}, a \in B^j,\ i \in A_{\mathrm{aux}}$,

$s \xrightarrow{a}_{1,j} s' \ \supset\ s \xrightarrow{a}_{2,j} s'$,

$s \xrightarrow{i}_{1,j} s' \ \wedge\ i \in Disabling^j \cup Enabling^j \ \supset\ s \xrightarrow{i}_{2,j} s'$,

$s \xrightarrow{i}_{1,j} s' \ \wedge\ i \notin Disabling^j \cup Enabling^j \ \supset\ s \xrightarrow{\epsilon}_{2,j} s'$.

C^j_2：以下を満たす最小の関係：

for $\forall s \in S, t \in S_{\mathrm{aux}}, a \in B^j,\ i \in A_{\mathrm{aux}}$,

$(s,a,t,i) \in C^j_1 \wedge i \in Disabling^j \cup Enabling^j \ \supset\ (s,a,t,i) \in C^j_2$.

□

以上の定義より，以下が成り立つことが容易にいえる．

補題 1 $\forall s,s',s'' \in S \cup S_{\mathrm{aux}}$, $a,a' \in Act$, $j \in [1,n]$ について以下が成り立つ．

(1) $a \in B^j$ ならば，$s \xrightarrow{a}_{1,j} s' \Leftrightarrow s \xrightarrow{a}_{2,j} s'$.

(2) $s \xrightarrow{\epsilon}_{2,j} s'$ ならば，ある $i^b_{s'} \in A_{\mathrm{aux}}$ が存在し，$s \xrightarrow{i^b_{s'}}_{1,j} s'$.

(3) $s' \in S_{\mathrm{aux}}$ ならば，任意の $i \in A_{\mathrm{aux}}$ について $s \not\xrightarrow{i}_{1,j} s'$.

(4) $s \in S_{\mathrm{aux}}$, $s \xrightarrow{a}_{1,j} s'$ かつ $s \xrightarrow{a'}_{1,j} s''$ ならば，$a = a' \in A_{\mathrm{aux}}$ かつ $s' = s''$.

□

3.3 ステップ (3)：L^j_{out3} を求める

ステップ (3) では，L_{out2} における ϵ 遷移を縮約する．図 3 では，ステップ (2) で得られた L^j_{out2} に対して ϵ 遷移を縮約した各成分 LTS L^j_{out3} を求めている．たとえば L^2_{out2} で状態 3 から 0 回以上の ϵ 遷移で到達可能な状態 3，5，0 を L^2_{out3} では 1 つの状態のように扱う．この状態の集合を L^2_{out2} における（状態 3 の）ϵ 閉包と呼び，$CL^2_\epsilon(3)$ で表す．L^2_{out3} は $CL^2_\epsilon(3)$ の中のどの状態にいるときでも，L^1_{out3} が i^a_1 を実行すれば，それに同期して状態 1 に遷移する．

定義 17 L^j_{out2} の任意の状態 $s \in S \cup S_{\mathrm{aux}}$ に対し，以下の条件を満たす最小の集合を $CL^j_\epsilon(s)$ と書く．

$$s \in CL^j_\epsilon(s),$$
$$s' \in CL^j_\epsilon(s) \ \wedge\ (s' \xrightarrow{\epsilon}_{2,j} s'') \ \supset\ s'' \in CL^j_\epsilon(s).$$

□

補題 2 $s \in S \cup S_{\mathrm{aux}}$, $s',s'' \in S$, $j \in [1,n]$, $a \in B^j$, $s' \in CL^j_\epsilon(s)$, $s'' \in CL_\tau(s')$, $s' \neq s$ かつ $s'' \xrightarrow{a}_{2,j} \cdot$ ならば，$s \in S_{\mathrm{aux}}$ かつ $CL^j_\epsilon(s) = \{s,s'\}$.

（証明）$s' \in CL^j_\epsilon(s)$ かつ $s' \neq s$ であるから，$\exists t \in CL^j_\epsilon(s)$ が存在し，$t \xrightarrow{\epsilon}_{2,j} s'$．補題 1 (2) より $t \xrightarrow{i^b_{s'}}_{1,j} s'$ である $i^b_{s'} \in A_{\mathrm{aux}}$ が存在．そのような任意の $t, i^b_{s'}$ について考える．

(1) $t \in S$ の場合．$\rightarrow_{1,j}$ の定義から $b \notin B^j$．しかし $a \in B^j$，$s'' \in CL_\tau(s')$ かつ $s'' \xrightarrow{a}_{2,j} \cdot$ なので，$i^b_{s'}$ は実行可能通知イベントとなり ϵ に変換されない．よって，仮定は矛盾．

(2) $t \in S_{\mathrm{aux}}$ の場合．補題 1 (2)，(3) より $u \xrightarrow{\epsilon}_{2,j} t$ となる u は存在しない（**図 5**）．$t \in CL^j_\epsilon(s)$ より，$t = s$．したがって $s \in S_{\mathrm{aux}}$．このとき補題 1 (2)，

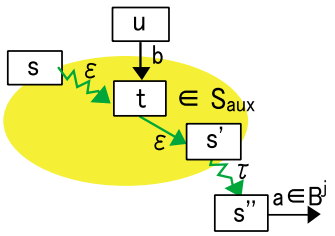

図 5 補題 2 の説明図（$t \in S_{\mathrm{aux}}$ の場合）
Fig. 5 Illustration of Lemma 2 ($t \in S_{\mathrm{aux}}$).

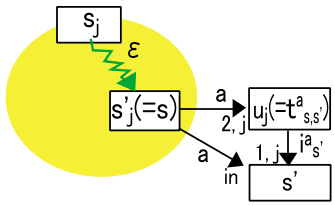

図 6 補題 3 の説明図
Fig. 6 Illustration of Lemma 3.

(4) より s からの遷移は $s \xrightarrow{\epsilon}_{2,j} s'$ のみ．また，$s'' \in CL_\tau(s')$ かつ $s'' \xrightarrow{a}_{2,j} \cdot$ であるから，実行不能通知イベントの条件から，仮にある $i \in A_{\rm aux}$ について $s' \xrightarrow{i}_{1,j} \cdot$ であったとしても，i は ϵ に変換されない．つまり，$s' \not\xrightarrow{\epsilon}_{2,j} \cdot$ である．以上のことから，$s \in S_{\rm aux}$ かつ $CL^j_\epsilon(s) = \{s, s'\}$ であるといえる． □

以下に LTS $L_{\rm out3}$ の定義を示す．

定義 18 $L^j_{\rm out2}$ $(j \in [1,n])$ に対し，$L^j_{\rm out3}$ を以下のとおり定義する．

$$
\begin{aligned}
&L^j_{\rm out3} = (A^j_2, S \cup S_{\rm aux}, \rightarrow_{3,j}, s_0, C^j_3)\\
&\text{where}\\
&\quad \text{for } \forall s, s' \in S \cup S_{\rm aux}, \alpha, a, i \in A^j_2,\\
&\quad\quad s \xrightarrow{\alpha}_{3,j} s' :\Leftrightarrow\\
&\quad\quad\quad t \xrightarrow{\alpha}_{2,j} s' \text{ for } \exists t \in CL^j_\epsilon(s),\\
&\quad\quad (s, a, s', i) \in C^j_3 :\Leftrightarrow\\
&\quad\quad\quad (t, a, s', i) \in C^j_2 \text{ for } \exists t \in CL^j_\epsilon(s).
\end{aligned}
$$

$L_{\rm out3} = (L^1_{\rm out3} \times \cdots \times L^n_{\rm out3}) \backslash A_{\rm aux}$ とする． □

以降，$L_{\rm in}$ と $L_{\rm out3}$ の双模倣等価性を示す．紙面の都合で証明を一部省略する．詳細は文献 5) を参照されたい．

補題 3 $s \in S$, $j \in [1,n]$, $a \in B^j$, $s_j, u_j \in S \cup S_{\rm aux}$, $s \in CL^j_\epsilon(s_j)$ かつ $s_j \xrightarrow{a}_{3,j} u_j$ ならば，ある $s' \in S$ が存在し，$u_j = t^a_{s,s'}$, $u_j \xrightarrow{i^a_{s'}}_{1,j} s'$ かつ $s \xrightarrow{a}_{\rm in} s'$ である．

（証明） $s_j \xrightarrow{a}_{3,j} u_j$ より，ある $s'_j \in CL^j_\epsilon(s_j)$ について $s'_j \xrightarrow{a}_{2,j} u_j$ である（**図 6**）．

(1) $s'_j \neq s_j$ のとき．補題 2 より，$CL^j_\epsilon(s_j) = \{s_j, s'_j\}$ かつ $s_j \in S_{\rm aux}$. $s \in CL^j_\epsilon(s_j)$ かつ $s \in S$ より $s = s'_j$. $s = s'_j \xrightarrow{a}_{2,j} u_j$, $a \in B^j$ であることと $\rightarrow_{1,j}$ の定義から，ある $s' \in S$ が存在し，$u_j = t^a_{s,s'} \xrightarrow{i^a_{s'}}_{1,j} s'$ かつ $s \xrightarrow{a}_{\rm in} s'$.

(2) $s'_j = s_j$ のとき．$s_j \xrightarrow{a}_{2,j} u_j$ であるから，実行不能通知イベントの定義より $s_j \not\xrightarrow{\epsilon}_{2,j} \cdot$ である．すなわち $CL^j_\epsilon(s_j) = \{s_j\}$. よって $s = s_j = s'_j$. 以降 (1) と同様． □

補題 4 $s \in S$, $j \in [1,n]$, $s_j, s' \in S \cup S_{\rm aux}$, $s \in CL^j_\epsilon(s_j)$ かつ $s_j \xrightarrow{i^\tau_{s'}}_{3,j} s'$ ならば，$s' \in S$ かつ $s \xrightarrow{\tau}_{\rm in} s'$ である．

（証明の方針）$\rightarrow_{3,j}$ の定義より，$t \xrightarrow{i^\tau_{s'}}_{2,j} s'$ である $t \in CL^j_\epsilon(s_j)$ が存在する．$i^\tau_{s'} \in Disabling^j$ のとき，$i^\tau_{s'} \in Enabling^j$ のとき，それぞれの場合について，補題 3 と同様に証明できる． □

定理 1 $L_{\rm in} \approx L_{\rm out3}$.

（証明） $\mathcal{R} \subseteq S \times (S \cup S_{\rm aux})^n$ を以下のように定義する．

$$
\begin{aligned}
&\mathcal{R} := \mathcal{R}_1 \cup \mathcal{R}_2\\
&\mathcal{R}_1 := \{(s, (s_1, \ldots, s_n)) \mid\\
&\quad s \in S,\ s \in CL^j_\epsilon(s_j) \text{ for } \forall j \in [1,n]\}\\
&\mathcal{R}_2 := \{(s, (t_1, \ldots, t_n)) \mid\\
&\quad s \in S,\ \exists i \in A_{\rm aux} - A_\tau,\ \forall j \in [1,n],\\
&\quad \text{if } i \in A^j_2 \text{ then } (t_j \xrightarrow{i}_{3,j} s_j) \wedge\\
&\quad s \in CL^j_\epsilon(s_j) \text{ for } \exists s_j \text{ else } s \in CL^j_\epsilon(t_j)\}.
\end{aligned}
$$

$\mathcal{R}$ が $(L_{\rm in}, L_{\rm out3})$ に対する弱双模倣関係であることを示す．以降では，$L_{\rm out3} = (A \cup \{\tau\}, (S \cup S_{\rm aux})^n, \rightarrow_3, (s_0, \ldots, s_0), C)$ とする．また，τ による遷移 $(s_1, \ldots, s_n) \xrightarrow{\tau}_3 (s'_1, \ldots, s'_n)$ がイベント $i \in A_{\rm aux}$ を隠蔽して得られたものであるとき，$(s_1, \ldots, s_n) \xrightarrow{\tau(i)}_3 (s'_1, \ldots, s'_n)$ と書く．

$\mathcal{R}$ が定義 7 の条件 (a) を満たす（$L_{\rm out3}$ が $L_{\rm in}$ を模倣できる）ことは容易に示せる．$\mathcal{R}$ が定義 7 の条件 (b) を満たす（$L_{\rm in}$ が $L_{\rm out3}$ を模倣できる）ことを示す．

(I) $(s, (s_1, \ldots, s_n)) \in \mathcal{R}_1$ の場合．$\mathcal{R}_1$ の定義より，任意の $j \in [1,n]$ について $s \in CL^j_\epsilon(s_j) \cdots (*1)$ である．$(s_1, \ldots, s_n) \xrightarrow{a}_3 (s'_1, \ldots, s'_n)$ である任意の $a \in A \cup \{\tau\}$, $(s'_1, \ldots, s'_n) \in (S \cup S_{\rm aux})^n$ について考える．

まず $a \neq \tau$ と仮定する．$(s_1, \ldots, s_n) \xrightarrow{a}_3 (s'_1, \ldots, s'_n)$ より，$a \in B^j$ である任意の $j \in [1,n]$（少なくとも 1 つ存在）について $s_j \xrightarrow{a}_{3,j} s'_j$. 補題 3 より，このような各 j に対して，$s'_j = t^a_{s,s'} \xrightarrow{i^a_{s'}}_{1,j} s'$ かつ $s \xrightarrow{a}_{\rm in} s'$ である $s' \in S$ が存在する．仮定 10 より各 j に対する s' はすべて等しい．以下，$(s', (s'_1, \ldots, s'_n)) \in \mathcal{R}_2$ を示す．任意の $j \in [1,n]$ について以下を考える．

(1) $a \in B^j$ のとき．$s'_j = t^a_{s,s'} \xrightarrow{i^a_{s'}}_{1,j} s'$ である．

(a) $i^a_{s'} \in A^j_2$ のとき．$s'_j = t^a_{s,s'} \xrightarrow{i^a_{s'}}_{3,j} s'$ かつ $s' \in CL^j_\epsilon(s')$ である．

(b) $i^a_{s'} \notin A^j_2$ のとき．$s'_j = t^a_{s,s'} \xrightarrow{\epsilon}_{2,j} s'$ なので $s' \in CL^j_\epsilon(s'_j)$ である．

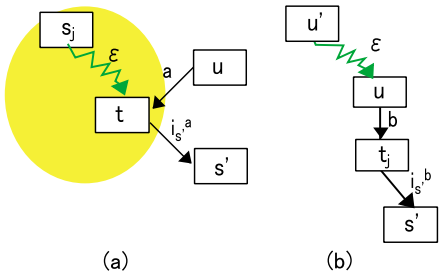

図 7 定理 1 の説明図
Fig. 7 Illustration of Theorem 1.

(2) $a \notin B^j$ のとき．$s_j = s'_j$ かつ $s \xrightarrow{i^a_{s'}}_{1,j} s'$ である．

(a) $i^a_{s'} \in A^j_2$ のとき．(∗1) より $s'_j = s_j \xrightarrow{i^a_{s'}}_{3,j} s'$ かつ $s' \in CL^j_\epsilon(s')$ である．

(b) $i^a_{s'} \notin A^j_2$ のとき．$s \xrightarrow{\epsilon}_{2,j} s'$ なので $s_j = s'_j$ と (∗1) より $s' \in CL^j_\epsilon(s'_j)$ である．

いずれも s' と s'_j は $\mathcal{R}_2$ の条件を満たす．

次に $a = \tau(i)$, $i \in A_{\mathrm{aux}}$ の場合について考える．$(s_1, \ldots, s_n) \xrightarrow{\tau(i)}_3 (s'_1, \ldots, s'_n)$ より，ある $k \in [1,n]$ が存在し，$i \in A^k_2$．また任意の $j \in [1,n]$ について，$i \in A^j_2$ ならば $s_j \xrightarrow{i}_{3,j} s'_j$，$i \notin A^j_2$ ならば $s_j = s'_j \cdots$(∗2) である．$i \in A_\tau$ かどうかによって場合分けする．

(1) $i \in A_\tau$ のとき．$i = i^\tau_{s'}$ とする．定義から，L^j_{out3} の任意の状態において $i^\tau_{s'}$ による遷移が存在すれば遷移先は s' に一意に決まる．よって，$i^\tau_{s'} \in A^j_2$ である j について $s_j \xrightarrow{i^\tau_{s'}}_{3,j} s'_j = s'$ である．このような j が少なくとも 1 つ存在することから，補題 4 と (∗1) より，$s \xrightarrow{\tau}_{\mathrm{in}} s'$ である．任意の $j \in [1,n]$ について，

- $i^\tau_{s'} \in A^j_2$ のとき，$s'_j = s'$ より $s' \in CL^j_\epsilon(s'_j)$.
- $i^\tau_{s'} \notin A^j_2$ のとき，$s \xrightarrow{\tau}_{\mathrm{in}} s'$ より $s \xrightarrow{\epsilon}_{2,j} s'$．これと (∗1)，(∗2) より，$s' \in CL^j_\epsilon(s'_j)$.

よって $(s', (s'_1, \ldots, s'_n)) \in \mathcal{R}_1$ である．

(2) $i \notin A_\tau$ のとき．$i = i^a_{s'}$ とする．定義 14，15 より，$i^a_{s'} \in A^j_2$ である $j \in [1,n]$ のうち，$a \in B^j$ である j が少なくとも 1 つ存在する．以下，このような j について考える．$s_j \xrightarrow{i^a_{s'}}_{3,j} \cdot$ より，ある $t \in CL^j_\epsilon(s_j)$ が存在し，$t \xrightarrow{i^a_{s'}}_{2,j} \cdot$．$a \in B^j$ より $t \in S_{\mathrm{aux}}$ (**図 7**(a))．補題 1 (2)，(4) より $t \not\xrightarrow{\epsilon}_{2,j} \cdot$ である．$s_j = t$ と仮定すると，$t \not\xrightarrow{\epsilon}_{2,j} \cdot$ かつ $t \neq s \in S$ なので (∗1) に矛盾．$s_j \neq t$ と仮定すると，補題 1 (2)，(3) より $u \xrightarrow{\epsilon}_{2,j} t$ となる u は存在しないので，$t \notin CL^j_\epsilon(s_j)$ となり矛盾．以上より $i \notin A_\tau$ はありえない．

(II) $(s, (t_1, \ldots, t_n)) \in \mathcal{R}_2 - \mathcal{R}_1$ の場合．$(t_1, \ldots, t_n) \xrightarrow{a}_3 (s'_1, \ldots, s'_n)$ である任意の $a \in A \cup \{\tau\}$, $(s'_1, \ldots, s'_n) \in (S \cup S_{\mathrm{aux}})^n$ について考える．

$(s, (t_1, \ldots, t_n)) \in \mathcal{R}_2 - \mathcal{R}_1$ より，ある $i^b_{s'} \in A_{\mathrm{aux}} - A_\tau$ が存在して，ある j と s' について $t_j \xrightarrow{i^b_{s'}}_{3,j} s'$ かつ $s \in CL^j_\epsilon(s') \cdots$(∗3) である．定義 14，15 より，このような j のうち $b \in B^j$ となる j が少なくとも 1 つ存在する．以下ではこのような j について考える (図 7(b))．

$t_j \xrightarrow{i^b_{s'}}_{3,j} s'$ より，ある $v \in CL^j_\epsilon(t_j)$ が存在し $v \xrightarrow{i^b_{s'}}_{2,j} s'$．$b \in B^j$ より $v \in S_{\mathrm{aux}}$．補題 1 (2)，(3) より $u \xrightarrow{\epsilon}_{2,j} v$ となる u は存在しないので，$v = t_j$．このとき定義から，ある $u \in S$ が存在し，$u \xrightarrow{b}_{2,j} t_j$ かつ $(u, b, t_j, i^b_{s'}) \in C^j_2$．よって $u \xrightarrow{b}_{3,j} t_j$ かつ $(u, b, t_j, i^b_{s'}) \in C^j_3$ となるので，$(t_1, \ldots, t_n) \xrightarrow{a}_3 \cdot$ となるのは $a = \tau(i^b_{s'})$ のときのみである．

以降では任意の $j \in [1,n]$ について考える．$(t_1, \ldots, t_n) \xrightarrow{\tau(i^b_{s'})}_3 (s'_1, \ldots, s'_n)$ であることから，$i^b_{s'} \in A^j_2$ ならば $t_j \xrightarrow{i^b_{s'}}_{3,j} s'_j$，$i^b_{s'} \notin A^j_2$ ならば $s'_j = t_j \cdots$(∗4) である．定義から，L^j_{out3} の任意の状態において $i^b_{s'}$ による遷移が存在すれば遷移先は s' に一意に決まるので，$i^b_{s'} \in A^j_2$ である j について $s'_j = s'$．(∗3) より $s \in CL^j_\epsilon(s'_j)$．また，$i^b_{s'} \notin A^j_2$ である j について $\mathcal{R}_2$ の定義から $s \in CL^j_\epsilon(t_j)$．よって (∗4) より $s \in CL^j_\epsilon(s'_j)$．以上より，$(s, (s'_1, \ldots, s'_n)) \in \mathcal{R}_1$．$s \stackrel{\epsilon}{\Rightarrow}_{L_{\mathrm{in}}} s$ すなわち $s \stackrel{\hat{\tau}}{\Rightarrow}_{L_{\mathrm{in}}} s$ より題意は示された．□

4. 手法の改良

4.1 インタリーブ領域の簡単化

並列システムで，複数の入力パラメータを必要とする処理を行うとき，そのパラメータの入力順を無視できることがよくある．たとえばシステムが状態 s において a，b という入力イベント系列および x という入力イベントを待っているとする．この仕様を表す L_{in} は**図 8**(a) になる．この LTS の状態 s から状態 t までの領域をインタリーブ領域と呼ぶことにする．a，b と x が別のプロセスで実行される場合，もし，インタリーブ領域を簡単化せずに L_{out3} に変換すると図 8(b) のようになる．図 8(c) のように簡単化できれば，システムの総状態数を削減できる．ここではこのインタリーブ領域を簡単化する方法について考察する．

任意の集合 Σ に対し，Σ の要素からなる有限の系列 $\langle a_1, \ldots, a_n \rangle$ を Σ 上の系列と呼ぶ．Σ 上の系列すべてからなる集合を Σ^* と書く．$\Sigma \subseteq Act$ の

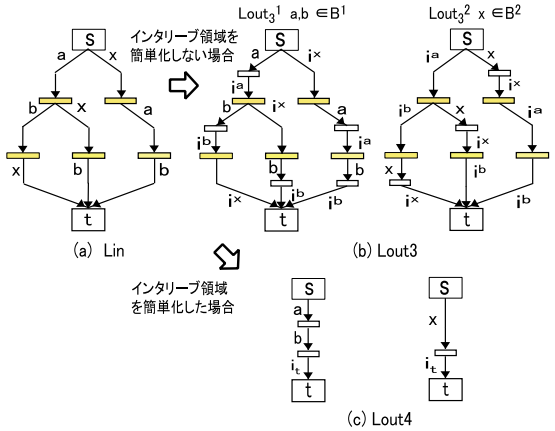

図 8 インタリーブ領域の簡単化
Fig. 8 Simplification of interleaving.

とき，Σ 上の系列をイベント系列と呼ぶ．任意の系列 α において，集合 A の要素である記号を残し他の記号を削除した系列を $\alpha \uparrow A$ と書く．たとえば $\langle a,x,b,c,c,y,x,z\rangle \uparrow \{a,b,c\} = \langle a,b,c,c\rangle$．空系列（長さ 0 の系列）を $\langle\rangle$ と書く．2 つの系列 α，α' の連接を $\alpha\hat{}\alpha'$ と書く．$\alpha = \langle a\rangle\hat{}\alpha'$ のとき，$head(\alpha) := a$，$tail(\alpha) := \alpha'$ と定義する．系列 α の長さを $|\alpha|$ と書く．$\forall k \in [1,|\alpha|]$ について，α の第 k 番の記号を $\alpha^{[k]}$ と書く．系列 $\alpha = \langle a_1,\ldots,a_m\rangle$ に対し，集合 $\{a_1,\ldots,a_m\}$ を $events(\alpha)$ と書く．

定義 19 遷移関係 $\rightarrow$ に対し，系列に対する遷移関係 $\rightarrow^* \subseteq S \times Act^* \times S$ を，以下を満たす最小の関係と定義する．

$$s \xrightarrow{\langle\rangle}{}^* s,$$
$$s \xrightarrow{a} s' \wedge s' \xrightarrow{\alpha'}{}^* t \wedge a = head(\alpha) \wedge \alpha' = tail(\alpha) \supset s \xrightarrow{\alpha}{}^* t.$$

□

定義 20 任意のイベント系列 α およびイベント系列の系列 Γ に対して，述語 $\alpha\ interleaves\ \Gamma$ を以下のように定義する[3]．

$\langle\rangle\ interleaves\ \langle\alpha_1,\ldots,\alpha_m\rangle :\Leftrightarrow$
$(\alpha_1 = \langle\rangle \wedge \cdots \wedge \alpha_m = \langle\rangle)$,
$(\langle a\rangle\hat{}\alpha')\ interleaves\ \langle\alpha_1,\ldots,\alpha_m\rangle :\Leftrightarrow$
for $\exists j \in [1,m]$, $\alpha_j \neq \langle\rangle \wedge head(\alpha_j) = a \wedge$
$\alpha'\ interleaves\ \langle\alpha_1,\ldots,tail(\alpha_j),\ldots,\alpha_m\rangle$.
ただし，$a \in Act$，$\alpha' \in Act^*$． □

定義 21 LTS $L = (A,S,\rightarrow,s_0,C)$ に対する状態遷移グラフ $G(L) = (V,E)$ は，

$$V := S,$$
$$E := \{(s,t) \in S \times S \mid s \xrightarrow{a} t \text{ for } \exists a \in A\}.$$

で定義される有向グラフである． □

定義 22 有向グラフ G におけるパス $\langle v_1,\ldots,v_m\rangle$ が単純であるとは，任意の相異なる $j,k \in [1,m]$ について，$v_j \neq v_k$ であることである． □

定義 23 有向グラフ $G = (V,E)$ および相異なる 2 頂点 $s,t \in V$ について，G の部分グラフ $G|_s^t$ を以下のように定義する．

$$G|_s^t := (V',E')$$
where
$V' := \{v \in V \mid t$ を含まないような s から v へのパスが存在，かつ s を含まないような v から t へのパスが存在 $\}$,
$E' := \{(u,v) \in E \mid u,v \in V'\}$.

□

定義 24 定義 9 の $L_{\rm in} = (A,S,\rightarrow_{\rm in},s_0,\emptyset)$，$n$ および B^j について考える．相異なる 2 状態 $s,t \in S$ に対し，系列の集合 $T_{s,t}$ を以下のように定義する．

$$T_{s,t} := \{\alpha \in A^* \mid s \xrightarrow{\alpha}{}^*_{\rm in} t\}.$$

$T_{s,t}$ が以下の条件を満たすとき，部分グラフ $G(L_{\rm in})|_s^t$ をインタリーブ領域と呼ぶ．

- あるイベント系列 $sq_1,\ldots,sq_n \in A^*$ が存在し，
  $T_{s,t} = \{\alpha \in A^* \mid \alpha\ interleaves\ \langle sq_1,\ldots,sq_n\rangle\}$.
ただし，任意の $j \in [1,n]$ について，
  $events(sq_j) \subseteq B^j$
  $\wedge\ events(sq_j) \cap B^k = \emptyset$ for $\forall k \neq j$.
- 状態遷移グラフ $G(L_{\rm in})$ が以下の条件を満たす．
  – 任意の $s' \in S$ について，t を含まないような s から s' へのパスが存在するならば，s' から t へのパスが存在．
  – 任意の $s' \in S$ について，s を含まないような s' から t へのパスが存在するならば，s から s' へのパスが存在．
  – s から t への任意のパスは単純． □

入力となる LTS を $L_{\rm in} = (A,S_{\rm orig},\rightarrow_{\rm in},s_0,\emptyset)$ とする．ある 2 状態 $src,to \in S_{\rm orig}$ について，部分グラフ $G(L_{\rm in})|_{src}^{to} = (V,E)$ がインタリーブ領域であると仮定する．このとき，以下のアルゴリズムを用いることができる．

(1) 新しいイベント β を導入し，src から to への遷移系列の集合を，β による 1 つの遷移に置き換える

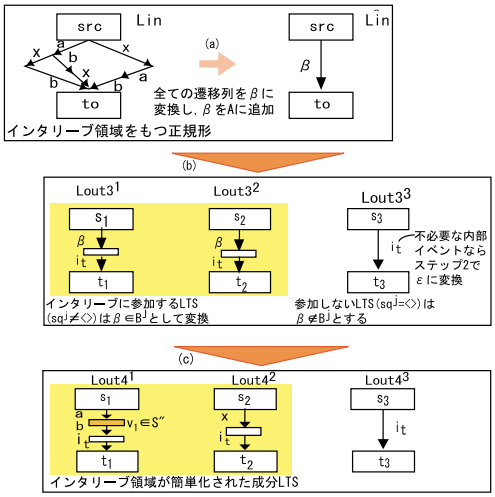

図 9 インタリーブ領域の簡単化の手順
Fig. 9 Method of the simplification.

(**図 9**(a)). こうして得られた LTS を $\hat{L_{\rm in}}$ とする．すなわち

$$\hat{L_{\rm in}} := (A \cup \{\beta\}, S, \rightarrow_{\hat{\rm in}}, s_0, \emptyset)$$
where
$$S := (S_{\rm orig} - V) \cup \{src, to\},$$
$$u \xrightarrow{a}_{\hat{\rm in}} u' :\Leftrightarrow (u \neq src \wedge u \xrightarrow{a} u') \vee (u = src \wedge a = \beta \wedge u' = to)$$

また，以下のように $\hat{B^j}$ を決める．

$$\hat{B^j} := \begin{cases} B^j \cup \{\beta\} & \text{if } sq_j \neq \langle\rangle, \\ B^j & \text{otherwise.} \end{cases}$$

(2) 3 章のアルゴリズムを $\hat{L_{\rm in}}$，$\hat{B^j}$ に適用し，$L^j_{\rm out3}$ を得る（図 9(b)）．ただし，$\beta \in \hat{B^j}$ である任意の $j \in [1,n]$ については，$L^j_{\rm out2}$ の定義において，同期イベント i^β_{to} が $Disabling^j \cup Enabling^j$ に含まれるものとして扱う（ϵ に変換しない）．このようにしても 3.3 節の補題・定理には影響しない．

(3) $L^j_{\rm out3}$ における β による遷移を，系列 sq_j による遷移系列に変換する．こうして得られた LTS を $L^j_{\rm out4}$ とする (図 9(c))．すなわち，$L^j_{\rm out3} = (A^j_3, S \cup S_{\rm aux}, \rightarrow_{3,j}, s_0, C^j_3)$ に対し，

$$L^j_{\rm out4} := (A^j_4, S \cup S_{\rm aux} \cup S_{\rm int}, \rightarrow_{4,j}, s_0, C^j_4)$$
where
$$A^j_4 := A^j_3 - \{\beta\},$$
$$S_{\rm int} := \{1, 2, \ldots, |sq_j| - 1\}.$$

読みやすさのため，$S_{\rm int}$ の要素 k を v_k と書く．

$$u \xrightarrow{\gamma}_{4,j} u' :\Leftrightarrow$$
$$(u \xrightarrow{\beta}_{3,j} \cdot \wedge\, u \xrightarrow{\gamma}_{3,j} u') \vee$$
$$(u \xrightarrow{\beta}_{3,j} \cdot \wedge\, \gamma = sq^{[1]}_j \wedge u' = v_1) \vee$$
$$(u = v_k \in S_{\rm int} \wedge k < |sq_j| - 1 \wedge \gamma = sq^{[k+1]}_j \wedge u' = v_{k+1}) \vee$$
$$(u = v_k \in S_{\rm int} \wedge k = |sq_j| - 1 \wedge \gamma = sq^{[k+1]}_j \wedge src \xrightarrow{\beta}_{3,j} u')$$
$$C^j_4 := C^j_3 - \{(u, \beta, t, i^\beta_{to}) \mid u \in S \cup S_{\rm aux}, t \in S_{\rm aux}, (u, \beta, t, i^\beta_{to}) \in C^j_3\}.$$

$L_{\rm out4} := (L^1_{\rm out4} \times \cdots \times L^n_{\rm out4}) \backslash A_{\rm aux}$ とし，これを出力する． □

上の定義で $S_{\rm int}$ はイベント β による遷移を系列 sq_j へ変換するときに追加される新しい状態の集合である．たとえば，$sq_1 = \langle a, b\rangle, sq_2 = \langle x\rangle$ とすると，$L^1_{\rm out4}$ では $|sq_1| = 2$ で $S_{\rm int} = \{v_1\}$ となり，$src \xrightarrow{a}_{4,1} v_1 \xrightarrow{b}_{4,1} u \xrightarrow{i^\beta_{to}}_{4,1} to$（ここで $src \xrightarrow{\beta}_{3,1} u$）となる．

このアルゴリズムによって得られる LTS $L_{\rm out4}$ は一般に，3 章のアルゴリズムで得られる LTS に較べて，状態数が少なく，同期イベントによる同期が必要な箇所も少なくなる場合が多い．すなわち，LTS を実現した際の動作効率が良いと考えられる．

定理 2 $L_{\rm in} \approx L_{\rm out4}$.

(証明の方針) $\mathcal{R} \subseteq (S \cup V) \times (S \cup S_{\rm aux} \cup S_{\rm int})^n$ を以下のように定義する．

$$\mathcal{R} := \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$$
where
$$\mathcal{R}_1 := \{(s, (s_1, \ldots, s_n)) \mid s \in S, s \in CL^j_\epsilon(s_j) \text{ for } \forall j \in [1,n]\}$$
$$\mathcal{R}_2 := \{(s, (t_1, \ldots, t_n)) \mid s \in S, \exists i \in A_{\rm aux} - A_\tau,\ \forall j \in [1,n],\ \text{if } i \in A^j_2 \text{ then } (t_j \xrightarrow{i}_{3,j} s_j) \wedge s \in CL^j_\epsilon(s_j) \text{ for } \exists s_j \text{else } s \in CL^j_\epsilon(t_j)\}.$$
$$\mathcal{R}_3 := \{(u, (u_1, \ldots, u_n)) \mid u \in V,\ src \xrightarrow{\alpha}{}^*_{\rm in} u \text{ for } \exists \alpha \in A^*,\ \text{for } \forall j \in [1,n], \alpha' = \alpha \uparrow B^j \wedge (s' \xrightarrow{\alpha'}{}^*_{4,j} u_j) \wedge src \in CL^j_\epsilon(s') \text{ for } \exists s'\}.$$

$\mathcal{R}$ が $(L_{\rm in}, L_{\rm out4})$ に対する弱双模倣関係であることを示すことができる． □

4.2 同期イベントの共通化

成分プロセス間で通信される同期イベントの種類を少なくできれば，通信する情報量が減少するので，通信がより効率的に行える．2 つの同期イベント i^a_s，i^b_t が次の条件を満たすとき，この 2 つの同期イベントを

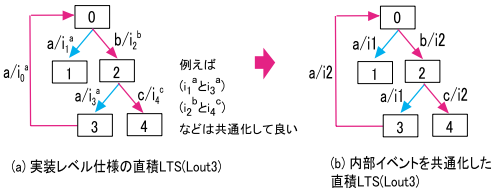

図 10 同期イベントの共通化
Fig. 10 Overloading of synchronization events.

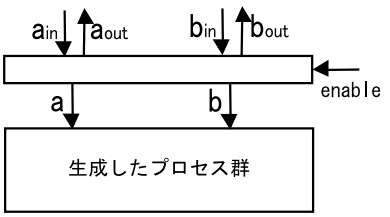

図 11 連続イベント制約が仮定できるシステム例
Fig. 11 An example which satisfies the contiguous event constraint.

共通化することが可能である．

条件：実装レベル仕様の直積 LTS (L_{out3}) のすべての状態 s に対して，$s \overset{a/i_s^a}{\not\rightarrow_3} \cdot$ または $s \overset{b/i_t^b}{\not\rightarrow_3} \cdot$．

ただし，$\rightarrow_3$ は L_{out3} の遷移関係を表す．また，$s \overset{a/i}{\rightarrow_3} s''$ は，外部イベント a と同期イベント $i \in A_{\mathrm{aux}}$ による遷移系列 $s \overset{a}{\rightarrow_3} s' \overset{\tau(i)}{\rightarrow_3} s''$ を表す．この遷移系列は各 L_{out3}^j の持つ連続イベント制約により不可分に実行される．たとえば**図 10**(a) のような L_{out3} では，状態 0 において $0 \overset{a/i_1^a}{\rightarrow_3} \cdot$ であるが，$0 \overset{a/i_3^a}{\not\rightarrow_3} \cdot$ である．一方，状態 2 においては $2 \overset{a/i_3^a}{\rightarrow_3} \cdot$ であるが，$2 \overset{a/i_1^a}{\not\rightarrow_3} \cdot$ である．他の状態からはこの 2 つの同期イベントに関係する遷移がないので，(i_1^a と i_3^a) は共通化できる．同様に (i_2^b と i_4^c) も共通化できる．この例で共通化可能なすべての同期イベントを共通化し，同期イベントにつけられている不要なラベルを消去すると図 10(b) のようになる．

5. 考　　察

以下では，連続イベント制約のない通常の遷移システムでプロセスの動作定義を与えるとした場合を考察する．これは，2.2 節の各定義において，連続イベント制約がつねに空であると仮定した場合に相当する．文献 2)，4)，6)，7) 等で行われているように以下の仮定をすることで，この場合でも提案アルゴリズムを適用することができる．

L_{in}，B^j $(1 \leq j \leq n)$ は仮定 10 および，以下の仮定を満たすとする．

仮定 25 任意の $s, s_1, s_2 \in S$，$a, b \in A - \{\tau\}$，$j \in [1, n]$ について，$s \overset{a}{\rightarrow_{\mathrm{in}}} s_1 \wedge s \overset{b}{\rightarrow_{\mathrm{in}}} s_2$ ならば，$a \in B^j \Leftrightarrow b \in B^j$． □

もし L_{in} が仮定 25 を満たさない場合，仮定 10 の場合と同様の方法で，仮定を満たしかつ元の L_{in} と弱双模倣等価であるような LTS を得ることができる[6),7)]．

上記の仮定を満たす L_{in}，B^j に対し，3 章のアルゴリズムを適用する．ただし，連続イベント制約の生成は行わない．このようにして得られた L_{out3} が L_{in} と弱双模倣等価であることを示せばよい．3.2，3.3 節の補題・定理のほとんどは，連続イベント制約の生成を行わなくても影響ない．影響があるのは，定理 1 の証明中の，(II) についてのみである．本論文では紙面の都合上，この証明を省略するが，仮定 25 と補題 1〜4 を用いることで L_{in} と L_{out3} の弱双模倣等価性を示すことができる．

上記のように，仮定 25 を満たさない L_{in} が与えられた場合，仮定を満たすように変形するためには内部イベント τ による遷移を余分に追加する必要がある．これは実装レベル仕様において，成分 LTS 間の A_τ の要素による同期を増やすこととなる．連続イベント制約を仮定できる状況では，連続イベント制約を用いたほうが，一般に成分 LTS 間の同期数を少なくできる．連続イベント制約を仮定できるようなシステムとしては**図 11** のような例が考えられる．仮定として LTS の各イベントによる遷移は時間 T_d 内に完了できるとする．このシステムは環境との通信にハンドシェイク機能を持ち，たとえば a_{in} から信号を入力した場合，その入力に対して受信完了通知または受信拒否通知を a_{out} から返す．環境は受信完了通知を受けたときのみ，そのイベントが発生したと解釈する．また，enable はクロック信号で，この信号が 1 なら環境から信号を受信でき，0 の間は受信できない．0 の期間は $2 \cdot T_d$ 以上続く信号とする．このシステムにおいて $enable = 1$ である時間内に複数の信号を入力した場合，そのうち 1 つの入力のみシステム内に通知し，その入力に対して受信完了通知を出力し，その他の入力には受信拒否通知を出力する．その後，少なくとも時間 $2T_d$ たってから，再び $enable = 1$ となる．たとえば環境からの入力 a_{in}，b_{in} のうち a_{in} が選択されたとすると，仮定より再び enable=1 となった時点では，システム内のプロセスはイベント a およびそれにともなう同期イベントによる遷移を完了している．したがって，このシステムでは連続イベント制約は満たされる．図 11 のような入力の選択機構はプライオ

リティエンコーダ[10] 等によって実現することが可能である．

6. あとがき

本論文では，要求仕様および，実装レベル仕様が持つべきプロセス数と各成分プロセスが用いる外部イベント集合（アルファベット）が与えられたとき，それらを満たす実装レベル仕様を導出する方法を示した．また，導出した直積 LTS と入力 LTS 間の弱双模倣等価性を証明した．さらに，インタリーブ領域を簡単化して総状態数を小さくする方法や，システム内部で用いられる同期イベントを共通化する方法を示した．

謝辞 通信プロトコル自動合成法の動向について種々ご教示いただいた，大阪大学大学院基礎工学研究科東野輝夫教授，岡野浩三講師に深謝いたします．

参考文献

1) Go, K. and Shiratori, N.: A Decomposition of a Formal Specification: An Improved Constraint-Oriented Method, *IEEE Trans. Softw. Eng.*, Vol.25, No.2, pp.258–273 (1999).
2) Higashino, T., Okano, K., Imajo, H. and Taniguchi, K.: Deriving Protocol Specifications from Service Specifications in Extended FSM Models, *The 13th IEEE ICDCS*, pp.141–148 (1993).
3) Hoare, C.A.R.: *Communicating Sequential Processes*, Prentice-Hall (1985).
4) Khoumsi, A., von Bochmann, G. and Dssouli, R.: On Specifying Services and Synthesizing Protocols for Real-time Applications, *Protocol Specification, Testing and Verification* (*PSTV XIV*), pp.185–200 (1994).
5) 喜家村奨：異なる並列度をもつ並列プロセス間の変換手法，奈良先端科学技術大学院大学修士論文 (2000).
6) Langerak, R.: Decomposition of Functionality: A Correctness Preserving LOTOS Transformation, *Protocol Specification, Testing and Verification* (*PSTV X*), pp.229–242 (1990).
7) 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，電子情報通信学会技術研究報告，Vol.IN91-110 (1991).
8) Saleh, K.: A Synthesis of Communications Protocols: An Annotated Bibliography, *ACM SIGCOMM Computer Communication Review*, Vol.26, No.5, pp.40–59 (1996).
9) 高橋 薫，神長裕明：仕様記述言語 LOTOS，カットシステム，pp.147–154 (1995).
10) Tokheim, R.L.（著）村崎憲雄，藤村宏一，青木正喜（訳）：ディジタル回路，pp.98–100, オーム社 (1995).

(平成 13 年 5 月 22 日受付)
(平成 13 年 10 月 16 日採録)

喜家村 奨（正会員）
平成 12 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．同年同大学博士後期課程入学，現在に至る．プロセス代数，ラベル付き遷移システムによる形式的仕様の記述およびその変換法についての研究に従事．

高田 喜朗（正会員）
平成 9 年大阪大学大学院基礎工学研究科博士後期課程修了．同年奈良先端科学技術大学院大学情報科学研究科助手．現在に至る．博士（工学）．ソフトウェアの仕様記述・解析手法，情報検索に関する研究に従事．

関 浩之（正会員）
昭和 62 年大阪大学大学院基礎工学研究科博士後期課程修了．工学博士．同年大阪大学基礎工学部情報工学科助手．同講師，助教授を経て，平成 6 年奈良先端科学技術大学院大学情報科学研究科助教授．平成 8 年同教授，現在に至る．形式言語理論，ソフトウェアの基礎理論に関する研究に従事．平成 9 年度情報処理学会論文賞受賞．