時制論理に基づくプロトコルの LOTOS 仕様の合成

安藤敏彦[†] 加藤 靖[†]
高橋 薫[††] 野口正一[†††]

プロトコル等，通信システムの仕様を曖昧なく記述するために，種々の形式記述技法が開発されている．これらの形式記述技法は数学的に厳密な取り扱いができる反面，記述が局所的であるために，仕様の大局的な振る舞いを陽に表現することができないという欠点がある．そのため，形式記述技法で記述された仕様が安全性，生存性等の時間的性質を満足しているかどうかを直観的に判断することは一般に困難である．一方，時系列上の性質を表現するために構成された論理体系が時制論理である．時制論理を用いると時間的性質を陽に表現することができ，大局的な振る舞いを理解しやすい．本論文では，形式記述技法の一つである LOTOS を取り上げ，その意味を表すラベル付き遷移システム上で拡張分岐時制論理を定義する．そして，この論理で記述された時間的性質から，それを満足する LOTOS 記述を導出する方法を提案する．さらに，より詳細な時間的性質を加えることで仕様を詳細化する方法も併せて提案する．これにより，時制論理を基礎とする段階的な仕様化が達成できる．また，時間的性質としてプロトコルの生存性を与えると，あるクラスのプロトコルの LOTOS 仕様が得られ，この方法がプロトコルの仕様を与える上で有効であることを示す．

Compositional LOTOS Specification of Protocol Based on Temporal Logic

Toshihiko Ando,[†] Yasushi Kato,[†] Kaoru Takahashi[††] and Shoichi Noguchi[†††]

FDTs (Formal Description Techniques) have been developed for specifying a communication system such as protocol explicity. Those FDTs can handle specifications mathematically, while it is not easy to describe explicitly the global behaviours of the specifications because of the locality of the description. Therefore it is difficult to understand intuitively whether a given specification satisfies the temporal properties, such as safety and/or liveness properties. By the way, there exists temporal logic which is the logical system to express properties on a time sequence. It is possible to explicity express temporal properties by using temporal logic, and it is also easy to understand global properties. We propose a method that can derive a specification described in LOTOS, one of the FDTs, from temporal properties based on EBTL (Extended Branching time Temporal Logic). EBTL is defined on labelled transition systems which give the semantics of LOTOS. Furthermore, we present a method by which it is possible to refine the specification by adding more detailed temporal properties. This method can achieve the stepwise development of a specification. Finally, we demonstrate that we can get a LOTOS specification of the protocol from liveness properties by applying this method to a protocol.

1. はじめに

プロトコル等，通信システムの仕様を記述する方法として形式記述技法 (Formal Description Technique, FDT) が用いられている，これは，自然言語に見られる曖昧さを避けるために開発されたもので，LOTOS, SDL, Estelle 等の FDT が国際標準化されている[1]．これら FDT を用いれば仕様を数学的に厳密に取り扱える．ところが，どの FDT も個々のアクションに対するプロセスの振る舞いを記述する形式を取るため，各時点での振る舞いは示すことができても，時間的に離れたアクション間の関係等，プロセスの時間的な振る舞いに関する全体的な性質を陽に記述することはできない．そのため，FDT で記述された仕様から時間的な性質を直接導出したり，直観的に理解することは困難である．

そのような FDT の局所性を補助する手段として，時制論理 (Temporal Logic) を利用することが考えられている[2]．時制論理は命題論理等の通常の論理体系

† 仙台電波工業高等専門学校
Sendai National College of Technology
†† 東北大学電気通信研究所
Research Institute of Electrical Communication, Tohoku University
††† 東北大学応用情報学研究センター
Research Center for Applied Information Sciences, Tohoku University

に，□（常に），◇（いつか）等の時間演算子を加え構成した論理体系である．時制論理は適用するモデルによって，線形時制論理 (Linear time Temporal Logic, LTL)，分岐時制論理 (Branching time Temporal Logic)[3]等と呼ばれる．時制論理を用いれば，安全性 (safety property)，生存性 (liveness property) 等の性質を陽に示すことができ，直観的に理解しやすい．時制論理を用いて FDT で記述された仕様が安全性，生存性を満足しているかどうかを検証する試みが行われている．例えば，有限状態機械と時制論理に基づいた SDL プロセスのモデルチェッカ[4]，LOTOS 仕様に対する時間的意味の構成的導出[5]等がそれである．また，これとは逆に，与えられた時間的性質を満たす仕様を導出することができるのであれば，仕様化の方法として有効である．しかし，一般に一つの時間的性質を満足する仕様は唯一ではなく，標準的な仕様の導出が望まれる．

本論文では，LOTOS 記述の意味であるラベル付き遷移システム (Labelled Transition System, LTS) から得られるアクション系列の集合をモデルとする拡張分岐時制論理 (Extended Branching time Temporal Logic, EBTL) を定義し，EBTL で記述された時間的性質を基に，これを満足する基本 LOTOS (Basic LOTOS) 仕様の構成法を提案する．LTS は LOTOS 記述に対応しており，LTS が得られるならば，それに対応する LOTOS 記述が導出できる．さらに，この方法をプロトコルに適用することによって，あるクラスの LOTOS 仕様を得ることができることを示す．

また，より詳細な時間的性質を加えることで仕様を詳細化する方法も提案する．これにより時制論理を基礎とした段階的な仕様化が達成できる．

本論文は次のように構成される，2章では LOTOS について簡単な説明を行い，以降の議論の準備として3章で LTS から得られるアクション系列の定義を行う．4章では時制論理の概略および EBTL の定義を行う．また，5章では時間的性質からの LOTOS 記述の構成法を与える．そして，この方法のプロトコルへの適用例を6章で示す．最後に，7章で結論を述べる．

2. LOTOS

LOTOS は OSI (Open Systems Interconnection) プロトコルを記述するためにプロセス代数を基に開発された FDT で，現在，国際標準化されている[6]．LOTOS では対象とするプロセスをブラックボックスとし，外部から観測できるイベント（アクション）の系列によって記述する．LOTOS において，データを扱わないものを特に基本 LOTOS (Basis LOTOS) と呼んでおり，本論文では，基本 LOTOS を扱うものとする．基本 LOTOS の構文の内，本論文で扱うものを**表1**に示す．

基本 LOTOS においてプロセスは次のように記述される．

process プロセス名 :=
　〈動作式〉
endproc

例えば，次のように記述することができる．

process P[a, c] :=
　hide b **in**
　a; (a; a; **stop** [] b; c; **stop**)
endproc

上の記述の中で，**hide** b **in** はアクション b が外部から観測できないことを宣言している部分である．LOTOS において，そのようなアクションのことを内部アクション (internal action) と呼び，**i** で表す．LOTOS で記述されるプロセスの意味は LTS で表される．LTS の方が扱いやすいため，ここでは LOTOS の記述は直接用いず，LTS で表現する．LOTOS 表現と LTS 表現は対応しており，LTS が求まれば，それに対応する LOTOS 表現が得られる．上の例の

表 1 基本 LOTOS の構文と意味
Table 1 Syntax and semantics of basic LOTOS.

オペレータ	構文	意味
Inaction	**stop**	プロセスの停止
Internal action	**i**; B	内部アクション **i** の実行
Observable action	α; B	外部アクション α の実行
Exit	**exit**	正常終了
Choice	B1 [] B2	B1 と B2 の内の可能なアクションの実行
Hiding	**hide** S **in** B	アクションリスト S に含まれるアクションの隠蔽
Process instantiation	$p[a_1, \cdots, a_n]$	プロセスのインスタンシエーション

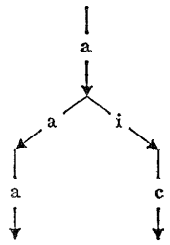


図 1 LTS による表現の例
Fig. 1 The example of an LTS expression.

LOTOS 表現に対する LTS は図1に示される．LTS の定義は次章で行う．なお，LOTOS についての詳細は文献6) を参照されたい．

3. LTS から導出されるアクション系列

本章では，LTS および，それから導出されるアクション系列を定義する．一般の LTS は非決定的で，一つの状態系列に対して複数のアクション系列が存在するが，本論文では決定的な LTS を扱う．ここで，決定的な LTS とは，任意の状態系列に対し，それに対応するアクション系列が必ずただ一つ決定される LTS を言う．決定的な LTS を扱うことは，LTS の合成を行う上で都合が良い．

[定義 1] LTS

LTS Sys は 4 項組 $\langle S, T, L, s_0\rangle$ から構成される．ここで，S は有限状態集合，T は S 上の 2 項関係で，T の元を枝と呼ぶ．$L: T \to Act$ は枝に対しアクションを与える写像である．ただし，Act はアクション全体の集合で，観測可能アクション，内部アクション **i**，正常終了アクション δ からなる．また，$s_i, s_j \in S$，$(s_i, s_j) \in T$，$\alpha \in Act$ に対し，$L((s_i, s_j)) = \alpha$ であるとき，$s_i \xrightarrow{\alpha} s_j$ と書く．$s_0 \in S$ は初期状態である．また，$\forall s' \in S$ に対して，$(s, s') \notin T$ であるような状態 s を Sys の葉と言う． □

[定義 2] 経路

LTS Sys $=\langle S, T, L, s_0\rangle$ の状態系列 $(s_i, s_{i+1}, \cdots, s_{i+n})$ を s_i から始まる経路と言い，s_i から始まる他の経路に真に含まれない経路全体の集合を $Path^{+i}$(Sys) と書く． □

[定義 3] 経路に付随するアクション系列

LTS Sys $=\langle S, T, L, s_0\rangle$ の経路 $x=(s_i, s_{i+1}, s_{i+2}, \cdots)$ に対し，アクション系列 $\sigma=(a_{i+1}, a_{i+2}, \cdots)$ が，$i+1 \leq \forall j$ に対して，$s_{j-1} \xrightarrow{a_j} s_j$ ならば，σ を x に付随するアクション系列と言い，$\sigma = seq(x)$ と書く．特に，s_0 から始まる経路に付随するアクション系列を Sys から導出されるアクション系列と言う．s_i から始まる経路に付随するアクション系列全体の集合を Seq^{+i}(Sys) と書く． □

決定的な LTS を扱っているため，Seq^{+i}(Sys) は Sys に対し，ただ一つ定まる．

[定義 4] 到達可能性

二つの状態 s_i，s_j に対し，s_i を始点，s_j を終点とするような有限状態系列 $x=(s_i, \cdots, s_j)$ が存在し，$\sigma = seq(x)$ ならば，σ によって s_i から s_j に到達可能であると言う．この時，$s_i \xrightarrow{\sigma} s_j$ と書く． □

[定義 5] 部分 LTS，幹，枝葉

LTS Sys′ $=\langle S', T', L', s_0'\rangle$ が Sys $=\langle S, T, L, s_0\rangle$ の部分 LTS であるとは，S′，T′ が各々 S，T の真部分集合であり，$L' = L|T'$ (L の T′ への制限)，および $\forall s' \in S'$ に対して，s′ は s_0' から到達可能であることを言う．特に，初期状態が s_0 である部分 LTS を幹と言い，葉を一つでも含むような s_0 を含まない部分 LTS を枝葉と言う． □

ここで定義した LTS の概観を図2に示す．LTS の状態および枝はそれぞれ有向グラフの頂点および辺で表され，枝にはアクションが付随している．ここで，s_0 は初期状態，s_4，s_{10} 等は葉である，LTS の経路には必ずただ一つのアクション系列が付随しており，例えば経路 $(s_0, s_1, s_2, s_3, s_4)$ にはアクション系列 (a, c, d, d) が対応する．また，Sys_1，Sys_2 はこの LTS Sys であるが，それぞれ Sys の幹，枝葉になっている．

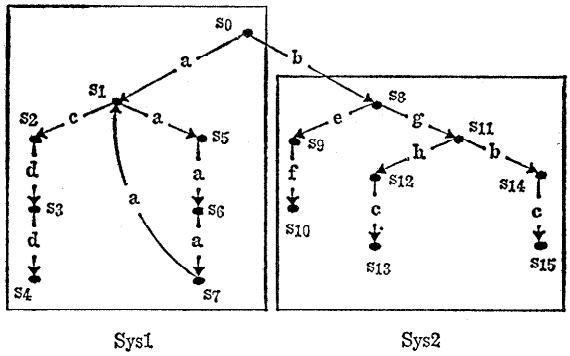


図 2 LTS の概観
Fig. 2 An overview of an LTS.

4. 時制論理

状態系列 $x=(s_0, s_1, s_2, \cdots)$ 上で満足される性質を時制論理を用いて表現することができる．時制論理は，様相論理 (Modal Logic) の一種であり，様相論理で

用いられる演算子 □ (must), ◇ (may) を時系列上で解釈したものである.

本論文では, LOTOS 記述を時系列としてその時間的性質を表現する必要があるが, LOTOS が外部観測可能なアクション (イベント) を基にしているので, 状態系列よりもアクション系列を扱うことの方が重要である. そこで, 本章では LTS から得られるアクション系列上で時制論理を構成する.

4.1 線形時制論理 (LTL)

本節では, LTS から導出されるアクション系列上での時間的性質を表現するために, アクション系列をモデルとする LTL を定義する. LTL は, 時間演算子 □ (always), ◇ (sometime), ○ (next), $\mathcal{U}$ (until), $\mathcal{W}$ (unless) を用いて構成される[5].

LTL の構文を次のように定義する.

[定義 6] Φ を原子命題全体の集合とする. LTL の論理式を次の規則によって構成する.

(1) $p \in \Phi$ ならば, p は論理式である.

(2) P を論理式とすると, ¬P も論理式である.

(3) P, Q を論理式とすると, P∧Q, P∨Q, P⇒Q, P≡Q, P⊕Q は論理式である.

(4) P を論理式とすると, □P, ◇P, ○P は論理式である.

(5) P, Q を論理式とすると, $P\,\mathcal{U}\,Q$, $P\,\mathcal{W}\,Q$ は論理式である. □

上の定義において, (1)～(3)は命題論理の規則であり, (4), (5)が時間演算子に関わる規則である. ここでは, アクションの系列をモデルとしているから, 原子命題にはアクションが対応する.

[定義 7] LTL のモデルはアクション系列 $\sigma=(a_1, a_2, a_3, \cdots)$ である. ここで, $a_i \in Act$ ($i \in \mathbf{N}$, $\mathbf{N}$ は自然数全体の集合), Act はアクション全体の集合である. □

また, LTL の論理式は, 充足関係 (satisfaction relation)⊨ によって, アクション系列上で次のように意味を与える.

[定義 8] アクション系列 $\sigma=(a_1, a_2, a_3, \cdots)$ をモデル, P, Q を論理式, $a \in Act$ とするとき, 充足関係⊨を σ の第 i アクションに対して次のように定義する.

$\sigma, i \models a$　　iff $a_i = a$

$\sigma, i \models \neg P$　　iff $\sigma, i \models P$ ではない

$\sigma, i \models P \wedge Q$　　iff $\sigma, i \models P$ かつ $\sigma, i \models Q$

$\sigma, i \models P \vee Q$　　iff $\sigma, i \models \neg(\neg P \wedge \neg Q)$

$\sigma, i \models P \Rightarrow Q$　　iff $\sigma, i \models \neg P \vee Q$

$\sigma, i \models P \equiv Q$　　iff $\sigma, i \models (P \Rightarrow Q) \wedge (Q \Rightarrow P)$

$\sigma, i \models P \oplus Q$　　iff $\sigma, i \models (P \wedge \neg Q) \vee (\neg P \wedge Q)$

$\sigma, i \models \Box P$　　iff $\forall j \geq i$ に対して, $\sigma, j \models P$

$\sigma, i \models \Diamond P$　　iff $\exists j \geq i$ に対して, $\sigma, j \models P$

$\sigma, i \models \bigcirc P$　　iff $\sigma, i+1 \models P$

$\sigma, i \models P\,\mathcal{U}\,Q$　　iff $\exists j \geq i$ に対して, ($\sigma, j \models Q$ かつ ($i \leq \forall k < j$ に対して, $\sigma, k \models P$))

$\sigma, i \models P\,\mathcal{W}\,Q$　　iff $\sigma, i \models P\,\mathcal{U}\,Q$ または $\sigma, i \models \Box P$ □

二つの時間演算子□, ◇は次のような相補性がある.

$\Box P \equiv \neg\Diamond\neg P$

$\Diamond P \equiv \neg\Box\neg P$

[注意 1] 同時に異なるアクションは起こり得ないので, 例えば, 異なるアクション a, b に対して,

$\sigma, i \models a \wedge b$

は成り立たない. □

4.2 拡張分岐時制論理 (EBTL)

一般に, 一つの LTS からは複数のアクション系列が得られる. そのため, すべてのアクション系列で, ある性質が成り立つ場合や, 一部のアクション系列では成り立つ性質が他のアクション系列では成り立たない場合が生じる. これを表現するため, 分岐するアクションの系列をモデルとする EBTL を構成する.

[定義 9] EBTL の論理式を次の規則によって構成する.

(1) P を命題論理の論理式とすると, P は論理式である.

(2) P を論理式とすると, **ALL** P, **some** P, **ALWAYS** P, **always** P, **SOMETIME** P, **sometime** P, **NEXT** P, **next** P は論理式である.

(3) P, Q を論理式とすると, P **UNTIL** Q, P **until** Q, P **UNLESS** Q, P **unless** Q は論理式である. □

次に, EBTL の意味を示す.

[定義 10] LTS $Sys=\langle S, T, L, s_0\rangle$ が与えられた時, EBTL のモデル $\mathcal{M}$ は 2 項組 $\langle S, Seq(Sys)\rangle$ からなる. ここで, $Seq(Sys)=\{\sigma \in S_{eq}{}^{+i}(Sys) \mid s_i \in S\}$ である. □

[定義 11] EBTL の充足関係 ⊨ を, $\mathcal{M}=\langle S, Seq(Sys)\rangle$ をモデルとし, LTL を用いて次のように定義する. ここで, $a \in Act$, $P, Q \in Prop$ ($Prop$ は EBTL の論理式全体の集合) とする.

$\mathcal{M}, s_i \models a$　　iff $(s_i, s_j) \in T$ なる s_j が

	ただ一つ存在し，s_i から始まる経路 x に対して，$seq(x)$, 1⊨a
$\mathcal{M}, s_i \models \neg P$	iff $(\mathcal{M}, s_i \models P)$ ではない
$\mathcal{M}, s_i \models P \wedge Q$	iff $\mathcal{M}, s_i \models P$ かつ $\mathcal{M}, s_i \models Q$
$\mathcal{M}, s_i \models$ **ALL** P	iff $\forall x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨P
$\mathcal{M}, s_i \models$ **some** P	iff $\exists x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨P
$\mathcal{M}, s_i \models$ **ALWAYS** P	iff $\forall x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨□P
$\mathcal{M}, s_i \models$ **always** P	iff $\exists x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨□P
$\mathcal{M}, s_i \models$ **SOMETIME** P	iff $\forall x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨◇P
$\mathcal{M}, s_i \models$ **sometime** P	iff $\exists x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨◇P
$\mathcal{M}, s_i \models$ **NEXT** P	iff $\forall x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨○P
$\mathcal{M}, s_i \models$ **next** P	iff $\exists x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨○P
$\mathcal{M}, s_i \models$ P **UNTIL** Q	iff $\forall x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨P$\mathcal{U}$Q
$\mathcal{M}, s_i \models$ P **until** Q	iff $\exists x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨P$\mathcal{U}$Q
$\mathcal{M}, s_i \models$ P **UNLESS** Q	iff $\forall x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨P$\mathcal{W}$Q
$\mathcal{M}, s_i \models$ P **unless** Q	iff $\exists x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨P$\mathcal{W}$Q

ただし，命題論理の他の演算子は定義8のそれと同様に定義するものとする． □

以下では，EBTL の論理式を時間的性質（temporal property）と呼び，LTS の時間的性質と言えば，初期状態 s_0 での時間的性質を指すものとする．さらに，時間的性質を記述するのに便利な演算子も定義しておく．

[定義12]

$\mathcal{M}, s_i \models$ **FREQ** P	iff $\forall x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨□◇P
$\mathcal{M}, s_i \models$ **freq** P	iff $\exists x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨□◇P
$\mathcal{M}, s_i \models$ P **BEFORE** Q	iff $\forall x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨P∧○◇Q
$\mathcal{M}, s_i \models$ P **before** Q	iff $\exists x \in Path^{+i}(Sys)$ に対して，$seq(x)$, 1⊨P∧○◇Q □

これらの演算子のうち，大文字で書かれたものはすべての系列で真となることを示し，小文字で書かれたものは真となる系列が存在することを示す．

4.3 生存性

プロセスの時間的性質で重要であるのが生存性である．生存性とは「将来良いことが起こるであろう」ことを示す性質である．例えば，

ALWAYS ((**freq** (データを送る))→(**sometime** (データを受け取る)))

：送信側がデータを送り続ければ，いつか受信側はデータを受け取る．

ALWAYS ((データを受け取る)⇒(**SOMETIME** (確認を返す)))

：受信側がデータを受け取れば，受信側は送信側に確認を返す．

がそうである．これはプロセスのどの状態でも成り立たなければならない性質，つまり，不変的（invariant）な性質である．EBTL では生存性を陽に表現することができ，一般に，次のように表現される．

ALWAYS(P⇒Q). (1)

5. 時間的性質からの LOTOS 仕様の導出

一般に，仕様に対する時間的性質は一意に決定できるが，与えられた時間的性質に対してそれを満足する

仕様を一意に決定することはできない．しかし，対象とするプロセスが複数のブロックから構成できるようなものとすれば，ある程度仕様の構造が決定される．本章では，与えられた時間的性質からそれを満足する LOTOS 仕様を導出する方法を示す．今後，LTS を図示する時，状態名は省略する．

5.1 時間演算子と LTS との対応

本節では，基本的な時間演算子に対する標準的な LTS を定める．ここで，標準的な LTS とは，与えられた時間的性質を満たし，アクション数が限定された LTS を言うものとする．これらは，LTS を合成するための部品として用いられる．

[定義 13] EBTL の演算子，および特定の論理式に対する標準的な LTS を**図 3** のように定義する．ここで，h はダミーアクションであり，空であるか，または他の LTS に置き換え得るものとする．a はアクション，P，Q は EBTL の論理式とする．また，＊はその性質を満たす有限なアクション系列であることを示す．□

5.2 LTS の接続と時間的性質の保存

本節では，LTS のシーケンシャル接続，および，それにより時間的性質がどう保存されるかについて述べる．シーケンシャル接続とは，一方の LTS の幹を他方の LTS の枝葉に重ねて新たな LTS を構成することである．

[定義 14] LTS のシーケンシャル接続

LTS $Sys_1=\langle S_1, T_1, L_1, s_{10}\rangle$ と $Sys_2=\langle S_2, T_2, L_2, s_{20}\rangle$ があり，Sys_1 と Sys_2 に共通するループを持たない部分 LTS $Sys'=\langle S', T', L', s_0'\rangle$ が存在し，Sys' は Sys_1 の枝葉かつ，Sys_2 の幹であるとする．このとき，

$$S=S_1\cup S_2$$
$$T=T_1\cup T_2$$
$$L: T\rightarrow Act;\ L((s_i, s_j))=\begin{cases}L_1((s_i, s_j)) & (s_i, s_j)\in T_1 \text{ の時}\\ L_2((s_i, s_j)) & (s_i, s_j)\in T_2 \text{ の時}\end{cases}$$
$$s_0=s_{10}$$

である $Sys=\langle S, T, L, s_0\rangle$ が作れるならば，Sys_1 に Sys_2 をシーケンシャルに接続できると言う．□

[注意 2]

定義 14 のように LTS を接続したとき，Sys の経路は元の 2 つの LTS の経路を接続して得られたものである．これらの経路は次の 3 通りに分類される．ただし，$St(x)$ は経路 x に含まれる状態全体の集合である．

- Sys_1 の経路で，共通部分 LTS Sys' の状態を含まないもの（タイプⅠ）．

$$Path^{(I)}(Sys)=\{x\in Path^{+0}(Sys)\mid St(x)\subseteq S_1-S'\}$$

- Sys_1，Sys_2 の経路を接続したものの内，共通部分以外の Sys_2 の状態を含むもの（タイプⅡ）．

$$Path^{(II)}(Sys)=\{x\in Path^{+0}(Sys)\mid St(x)\cap(S_2-S')\neq\emptyset\}$$

- 共通部分 LTS Sys' の葉を終点とする経路（タイプⅢ）．

$$Path^{(III)}(Sys)=Path^{+0}(Sys)-Path^{(I)}(Sys)-Path^{(II)}(Sys)$$

これらの経路に付随するアクション系列も各々のタイプに分類される．また，Sys_1 の経路も次の 2 通りに分類される．

- 共通部分を含まないもの（タイプⅠ）．

$$Path^{(I)}(Sys_1)=\{x\in Path^{+0}(Sys)\mid St(x)\subseteq S_1-S'\}$$

- 共通部分を含むもの（タイプⅡ，Ⅲ）．

$$Path^{(II,III)}(Sys_1)=Path^{+0}(Sys_1)-Path^{(I)}(Sys_1)$$

例えば，**図 4** で，Sys_1，Sys_2 をシーケンシャルに接

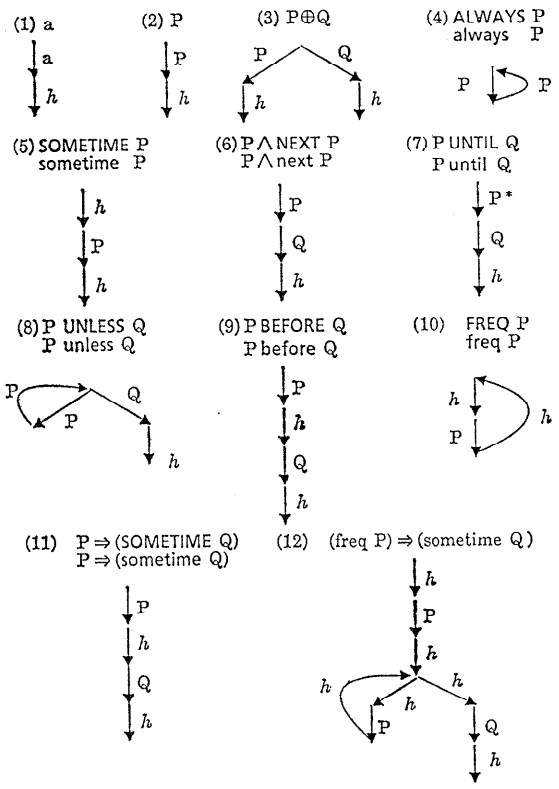


図 3 EBTL の演算子に対応する標準的な LTS
Fig. 3 The standard LTSs corresponding to EBTL operators.

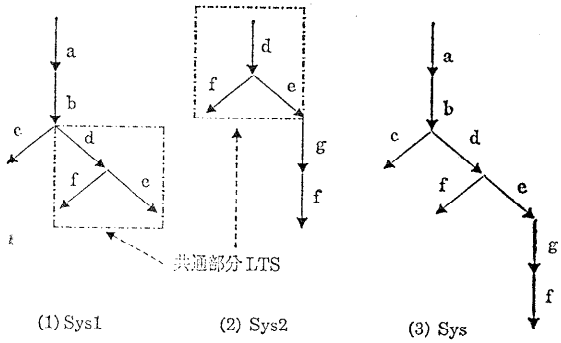


図 4 LTS のシーケンシャル接続
Fig. 4 Sequential connection of LTSs.

続すると，Sys が得られる．この時，Sys から導出されるアクション系列は σ_1=(a, b, c), σ_2=(a, b, d, e, g, f), σ_3=(a, b, d, f) であるが，このうち，σ_1 はタイプⅠ，σ_2 はタイプⅡ，σ_3 はタイプⅢである． □

次に，LTS をシーケンシャル接続した時の時間的性質の保存について述べる．まず，その様子を次の例に示す．

［例 1］ 図 4 で，Sys_1 では時間的性質 P_1=**SOMETIME** b が成り立ち，また，Sys_2 では，P_2=**SOMETIME** f が成り立つ．合成された LTS Sys では P_1 は成り立つが，P_2 は成り立たず，その代わり **sometime** P_2 が成り立つ． □

例 1 のように，LTS を接続した場合，元の時間的性質が保存される保証はない．しかし，ある条件の下では時間的性質が保存される．そこで，時間的性質を保存したまま LTS を合成できる条件を与えるため，以下の性質を定義する．

［定義 15］ 部分性，全体性

LTS Sys=⟨S, T, L, s_0⟩ が与えられており，モデル $\mathcal{M}$=⟨S, s_{eq}(Sys)⟩に対して，$\mathcal{M}$, s_0⊨P であるとする．このとき，ある Seq^{+0}(Sys) の真部分集合Xが存在し，$\mathcal{M}'$=⟨S, $s_{eq^{+0}}$(Sys)−X⟩ に対して，$\mathcal{M}'$, s_0⊨¬P ならば，P は $\mathcal{M}$ に対して部分的であると言い，PはX に依存すると言う．また，P が $\mathcal{M}$ に対して部分的でなければ，P は $\mathcal{M}$ に対して全体的であると言う． □

部分的な時間的性質は，複数の経路の内，一部の経路のみで決定されるようなものである．また，P=*op* P′ (*op* は大文字の演算子) の形の時間的性質は全体的である．

［定義 16］ 有界性，恒真性

LTS Sys=⟨S, T, L, s_0⟩ が与えられており，モデル $\mathcal{M}$=⟨S, s_{eq}(Sys)⟩ に対して，$\mathcal{M}$, s_0⊨P, (P=*op* P′, *op* は EBTL の演算子) であるとする．このとき，$\mathcal{M}$, s_0⊨**SOMETIME** (**ALWAYS** (¬P′)) ならば，P は $\mathcal{M}$ に対して有界的であると言う．また，$\mathcal{M}$, s_0⊨**ALWAYS** P′ ならば，P は $\mathcal{M}$ に対して恒真的であると言う． □

図 2 の LTS では，時間的性質 **always** a, **sometime** b が成り立つ．**always** a の成り立つアクション系列はただ一つであるので部分的である．また，アクション b はある時間以降起こることはないので，**sometime** b は有界的である．

次の命題 1 は，LTS の合成と，それに伴う時間的性質の保存に関する性質である．

［命題 1］ LTS Sys_1=⟨S_1, T_1, L_1, s_{10}⟩ に Sys_2=⟨S_2, T_2, L_2, s_{10}⟩ をシーケンシャルに接続して Sys=⟨S, T, L, s_0⟩ が得られたとする．このとき，$\mathcal{M}_1$=⟨S_1, *Seq*(Sys)⟩, $\mathcal{M}_2$=⟨S_2, s_{eq}(Sys_2)⟩ に対して $\mathcal{M}_1$, s_{10}⊨P, $\mathcal{M}_2$, s_{20}⊨Q であるとする．$\mathcal{M}$=⟨S, s_{eq}(Sys)⟩ とすると，次のいずれかの条件の下で，$\mathcal{M}$, s_0⊨P∧(**sometime** Q) が成り立つ．

(1) P=*op* P′ (*op* は EBTL の演算子) であり，P が $\mathcal{M}_1$ に対して有界的，かつ ¬P′ が $\mathcal{M}_2$ に対して恒真的である．

(2) P が $\mathcal{M}_1$ に対して部分的であり，$Path^{(I)}$(Sys_1) に依存する．

(3) P が $\mathcal{M}_1$ に対して恒真的であり，かつ $\mathcal{M}_2$ に対しても恒真的である．

(証明) **sometime** Q が成り立つのは明らか．(1) ¬P′ が $\mathcal{M}_2$ に対して恒真的であるから，P は $\mathcal{M}$ に対しても有界的である．したがって，P が成り立つ．(2) P は $Path^{(II,III)}$(Sys_1) には依存しないから，$Path^{(II)}$(Sys) や $Path^{(III)}$(Sys) に依存しない．一方，$Path^{(I)}$(Sys)=$Path^{(I)}$(Sys_1) であるので，P は $Path^{(I)}$(Sys) において成り立つ．したがって，P が成り立つ．(3) P は $\mathcal{M}_1$, $\mathcal{M}_2$ に対して恒真的であるから，$\mathcal{M}$ に対しても恒真的である．したがって，P が成り立つ． □

命題 1 より次の命題 2 が導かれる．

［命題 2］ 命題 1 (3) において，Q が特に **ALWAYS** (Q_1⇒Q_2) の形をしており，Q が Sys_1 でも成り立つならば，Sys において P∧Q が成り立つ． □

［注意 3］ 命題 2 は，生存性を矛盾しないように与えれば，それらが保存されるように LTS を合成できることを意味する． □

命題 1 および 2 は複数の時間的性質を満足するような LTS を合成する原理となる．

5.3 時間的性質から LOTOS の仕様の導出

以下では，時間的性質から LOTOS 仕様を導出するための構成法を示す．ここで次の記法を用いる．LTS Sys にダミーアクション h を除き陽に含まれるアクション全体の集合を Act(Sys)，Sys の幹全体の集合を $Root$(Sys) とする．また，時間的性質 P に h を除き陽に含まれるアクション全体の集合を Act (P) とする．

[構成法1] 時間的性質に基づく LOTOS 仕様の合成

(1) 性質 $P_1, \cdots, P_n$ を与える．ただし，これらの性質は命題1のいずれかの条件を満足するものとし，P_1は全プロセスの初期状態に関わるものとする．

(2) 定義13に従って，P_i $(1 \leq i \leq n)$ を満足する標準的 LTS Sys_i を求める．ただし，$Act(Sys_i) = Act(P_i)$ であるものとする．

(3) Sys_i と Sys_j $(1 \leq j \leq n, j \neq i)$ の共通部分 LTS を $Sys_{(i,j)}$ とする．ただし，$Act\ (Sys_{(i,j)}) \neq \emptyset$．ここで，$2 \leq i \leq n$ の Sys_i に対して，$\forall j \neq i$, $Sys_{(i,j)} \notin Root(Sys_i)$ であるならば(1)に戻り性質を与え直す．

(4) (3)で求めた $Sys_{(i,j)}$ が重なるよう，Sys_i と Sys_j とをシーケンシャル接続する．

(5) プロセスの動作が有限ならば，未接続の終端すべてに，プロセスの終了を示す特別のアクション δ を付け加える．また，動作が無限ならば，終端を Sys_1 の根に接続する．

(6) ループに分岐する h を内部アクション **i** で置き換える．必要ならば，その他の h アクションを **i** で置き換えてもよい．最後に，残りの h アクションを消去する．ここで得られた LTS を Sys とする．Sys が求める LTS である．

(7) LTS Sys を LOTOS 記述へ変換する． □

構成法1の(3)では，複数の小さな LTS をシーケンシャルに接続できるものだけを考えている．構成法1の(6)においてループに分岐する h (例えば，図3(12)) を内部アクションで置き換えるのは，このようなループがプロトコルにおいてはデータ等の再送を意味し，その原因は媒体の障害等観測不可能なアクションによると考えられるからである．

LTS から LOTOS 記述への変換法を変換法1に示す．

[変換法1] LTS から LOTOS 記述への変換

(1) 全体のプロセス名を $Proc_1$ とする．また，h アクションから置き換えられた内部アクションを，各々 $int_1, int_2, \cdots, int_p$ とし，合流地点に各々サブプロセス名 $Proc_2, \cdots, Proc_q$ をつける．

(2) **process** $Proc_1[\Delta]$:=

を生成する．Δ には，Act (Sys) のアクションのリストを列記する．根のアクションから合流地点までに内部アクションがあれば，次に

hide Δ_{int} **in**

を挿入する．ここで，Δ_{int} は合流地点までに出現する内部アクションのリストである．

(3) 根のアクションから順に次の手続きを行う．

(3.1) アクション系列が分岐しなければ，そのアクション系列をプレフィクス (;) を使って列記する．

(3.2) アクションが分岐すれば，分岐してできるアクション系列をチョイス ([]) で結ぶ．

(3.3) アクション系列が終端に達したら，それが δ アクションならば

exit

で置き換え，それ以外であれば，最後のアクションに

; **stop**

を付け加える．また，合流地点に達したら，

; (対応するサブプロセス名)

を付け加える．

(3.4) 最後に，

where

endproc

とする．

(4) 各サブプロセスは $Proc_1$ と同様に記述し，$Proc_1$ の **where** と **endproc** の間に挿入する． □

構成法1の例を示す．

[例2] 二つの性質 P_1 = **ALWAYS** ((**freq** a)⇒(**sometime** b))，P_2 = **ALWAYS** (b⇒(**SOMETIME** c)) を満足する LTS を合成する．

(1) 各時間的性質に対する LTS を求める (**図5**の(1)，(2))．

(2) 両者に共通する部分 LTS を求める．

(3) 二つの LTS を接続する (図5の(3))．

(4) ループに分岐する h アクションを内部アクションに置き換え，その他の h アクションを消去する (図5の(4))．

(5) LOTOS 記述を求める (**図6**)． □

図5は命題2が成立することを示している．

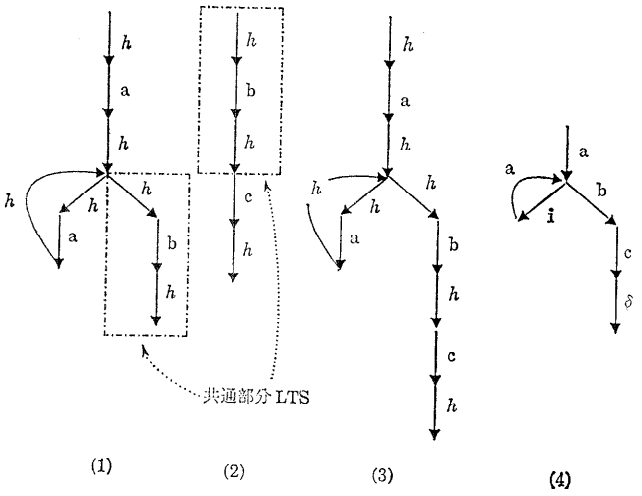


図 5 構成法の適用例
Fig. 5 An application example of the construction method.

```
process Proc1[a, b, d, e, c] :=
  a ; Proc2[a, b, d, e, c]
where
  process Proc2[a, b, d, e, c] :=
    hide int1 in
        int1 ; a ; Proc2[a, b, d, e, c]
      [] b ; d ; e ; c ; exit
    endproc
endproc
```

図 6 時間的性質 P_1, P_2 を満足する LOTOS 記述
Fig. 6 The LOTOS description satisfying the temporal properties P_1 and P_2.

5.4 段階的構成法

次に，構成法1を基本とした LOTOS 仕様の段階的な構成法を示す．より多くの時間的性質を付加し，構成法1の過程を繰り返せば，より詳細な仕様が得られる．すなわち，最初はプロセスの大まかな性質を与え，その後の段階では，より詳細な性質を与えるというように，段階的な仕様化を達成することができる．追加する性質は，前段階の性質に現れるアクションとアクションの間を補うように与える．

[構成法2] 時間的性質に基づく LOTOS 仕様の段階的構成法

(1) 性質 $P_1^{(1)}, \cdots, P_n^{(1)}$ から，構成法1に従って，LTS $Sys^{(1)*}$ が構成されているものとする．ただし，ダミーアクション h を消去せず残してある．

(2) 性質 $P_1^{(1)}$ に対し，性質 $P_{1,1}^{(2)}, \cdots, P_{1,n_1}^{(2)}$ を与える．これらの性質は $P_1^{(1)}$ に現れるアクションとアクションの間を補うものとする．

(3) 性質 $P_{1,1}^{(2)}, \cdots, P_{1,n_1}^{(2)}$ を満たす標準的な LTS を求め，この LTS を $Sys_1^{(2)}$ とする．

(4) $Sys^{(1)*}$ において，$P_1^{(1)}$ に相当する LTS $Sys_1^{(1)}$ を $Sys_1^{(2)}$ に置き換える．

(5) $P_i^{(1)}$ $(2 \leq i \leq n)$ について $P_1^{(1)}$ と同様に(2)～(4)を行う．

(6) プロセスの動作が有限ならば，未接続の終端すべてに，プロセスの終了を示す特別のアクション δ を付け加える．また，動作が無限ならば，終端を $Sys^{(1)*}$ の根に接続する．ここで得られる LTS を $Sys^{(2)*}$ とする．

(7) 必要であれば，$Sys^{(2)*}$ を $Sys^{(1)*}$ とみなして(1)～(6)を繰り返し，LTS $Sys^{(3)*}$ を求める．同様に，必要であれば，$Sys^{(4)*}, Sys^{(5)*}, \cdots$ を求める．

(8) $Sys^{(p)*}$ $(p \geq 2)$ において，ループに分岐する h を内部アクション **i** で置き換える．必要ならば，その他の h アクションを **i** で置き換えてもよい．最後に，残りの h アクションを消去する．このようにして得られた LTS を $Sys^{(p)}$ とする．

(9) LTS $Sys^{(p)}$ $(p \geq 2)$ を LOTOS 記述に変換する．

□

構成法2の(3)は，$Sys^{(p)*}$ に現れる h を追加した時間的性質に対する LTS に置き換える作業である．すなわち，一つのサブプロセスをさらに小さなサブプロセスでシーケンシャルに構成することに相当する．従って，構成法2は入れ子構造を持つプロセスの仕様化に有効である．また，構成法2によって，レベルの違った LTS の列 $Sys^{(1)}, Sys^{(2)}, \cdots$ が得られるが，これらの LTS の関係を次の命題3に示す．

[命題3] 生存性を時間的性質として与え，構成法2を行い，$Sys^{(1)}, Sys^{(2)}, \cdots$ が得られたとする．この時，LTS S が満足する性質の全体の集合を $Prop(S)$ と書くとすると，$m<n$ ならば，$Prop(Sys^{(m)}) \subseteq Prop(Sys^{(n)})$ である． □

[例3] 例2で与えた時間的性質に，新たな性質を付加し，仕様をより詳細にする．付加する条件は，$P_{2,1}$=**ALWAYS** (b⇒(**SOMETIME** d)), $P_{2,2}$=**ALWAYS** (d⇒(**SOMETIME** e)), $P_{2,3}$=**ALWAYS** (e⇒(**SOMETIME** c)) である．

(1) 付加する性質に対する標準的な LTS を求める．(**図7**の(1))

(2) 図5の(3)の LTS において，P_2 に対応する部

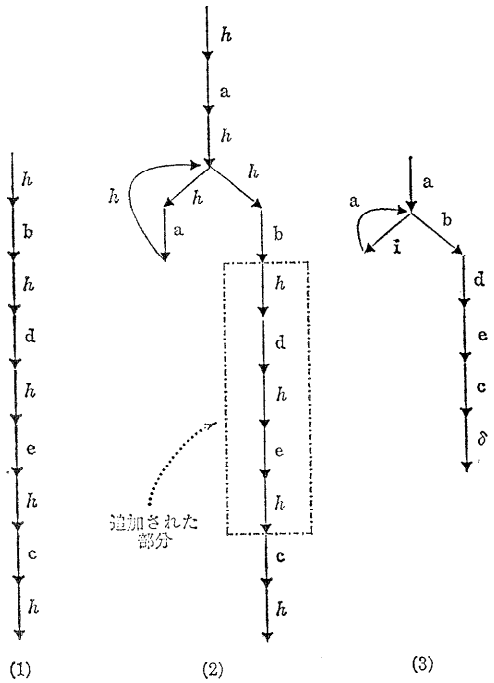


図 7 段階的構成法の適用例
Fig. 7 An application example of the stepwise construction method.

```
process Proc1[a, b, c] :=
  a ; Proc2[a, b, c]
where
  process Proc2[a, b, c] :=
    hide int1 in
        int1 ; a ; Proc2[a, b, c]
      [] b ; c ; exit
  endproc
endproc
```

図 8 時間的性質 $P_{2,1}$, $P_{2,2}$, $P_{2,3}$ を付加して得られる LOTOS 記述
Fig. 8 The LOTOS description obtained by adding the temporal properties $P_{2,1}$, $P_{2,2}$ and $P_{2,3}$.

分の LTS を上の LTS に置き換える（図7の(2)）.

(3) 分岐する h アクションを内部アクションに変え，有限動作であるから，終端に δ アクションを付け加え，余分な h アクションを消去する（図7の(3)）.

(4) この LTS を LOTOS 記述に変換する（**図8**）.

□

6. プロトコルへの適用

本章では，構成法 1，2 のプロトコルへの適用を行う．ここで扱うプロトコルはアブラカダブラプロトコル (Abracadabra Protocol) の簡略版である[1]．これは，送信側 (Sender) と受信側 (Receiver) との間で，信頼度の低い媒体 (Medium) を用いて一方向通信を行うためのプロトコルである．このプロトコルの生存性を時間的性質として与え，構成法1を用いて全体の振舞いを示す仕様を導出し，さらに，構成法2を用いて，より詳細な仕様を導出する．本章ではこの適用例を通して，この方法が実際的なプロトコルに対しても有効であることを示す．

6.1 アブラカダブラプロトコル

アブラカダブラプロトコルは，コネクションの確立，データの転送，コネクションの切断の三つのフェーズを持つ．データ等が媒体で欠落した場合は再送で補うようにしている．

6.1.1 各フェーズの概要

(1) コネクション確立フェーズ (Connection Phase)
送信側が受信側に接続要求 (CR) を送り，送信側が受信側から接続確認 (CC) を受け取ると，コネクションが確立され，データ転送フェーズに移る．CR, CC の欠落は再送によって復旧させる．切断要求 (DR) の送信はいつでも可能であり，DR が送られると，コネクション切断フェーズに移る（**図9**）.

(2) データ転送フェーズ (Data Transfer Phase)
送信側が受信側にデータを転送 (DT) し，受信側は送信側に受信確認 (AK) を返す．DT, AK の欠落は再送によって復旧させる．切断要求 DR の送信はいつでも可能であり，DR が送られると，切断フェーズに移る（**図10**）.

(3) コネクション切断フェーズ (Disconnection Phase)
送信側が受信側に切断要求 DR を送り，送

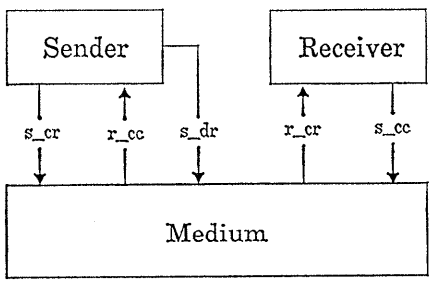


図 9 コネクション確立フェーズの概念図
Fig. 9 An overview of the Connection Phase.

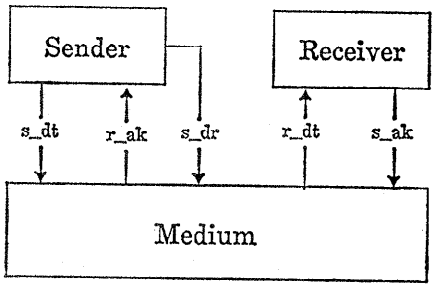


図 10 データ転送フェーズの概念図
Fig. 10 An overview of the Data Transfer Phase.

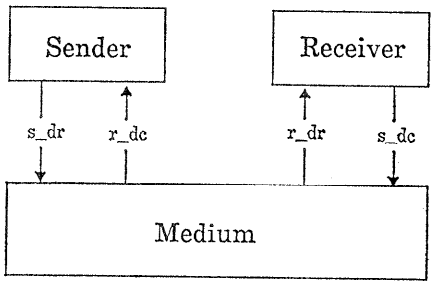


図 11 コネクション切断フェーズの概念図
Fig. 11 An overview of the Disconnection Phase.

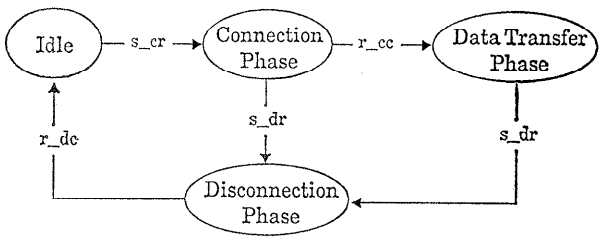


図 12 各フェーズ間の関係
Fig. 12 Relation among the Phases.

信側が受信側から切断確認 (DC) を受け取ると，コネクションが切断される．その後，コネクション確立フェーズに移る (**図 11**)．

上の 3 つのフェーズは**図 12** のように関連づけられる．Idle は初期状態である．

6.1.2 プロセス全体としての性質

プロセス全体の性質として，フェーズの遷移に関わる性質を与える．これらは特定のアクションが起これば次のフェーズに移るであろうことを示しているので，生存性である．全体的な時間的性質を次に示す．

(1) **ALWAYS** (s_cr⇒ ((**SOMETIME** (r_cc⊕s_d))))
：[コネクション確立フェーズ] 送信側が CR を送れば，いつか送信側は CC を受け取るか，または，DR を送る．

(2) **ALWAYS** (r_cc⇒(**sometime** s_dr))
：[データ転送フェーズ] 送信側が CC を受け取ればデータ転送フェーズに入り，送信側が DR を送るまではデータ転送フェーズに留まる．

(3) **ALWAYS** (s_dr⇒(**SOMETIME** r_dc))
：[コネクション切断フェーズ] 送信側が DR を送れば，いつか必ず送信側は DC を受け取り，コネクションが切断される．

(4) **freq** s_cr
：このプロセスは無限動作する．

6.1.3 各フェーズの詳細な性質

次に，各フェーズの詳細な性質を示す．

(1′) コネクション確立フェーズ

(1′.1) **ALWAYS** ((**freq** s_cr)⇒((**sometime** (r_cc⊕s_dr))))
：送信側が何度も CR を送れば，いつかは受信側に CR が受け取られる．または，送信側が DR を送る．

(1′.2) **ALWAYS** (r_cr⇒(**SOMETIME** (s_cc⊕s_dr))))
：受信側が CR を受け取れば，受信側は CC を返す．または送信側が DR を送る．

(1′.3) **ALWAYS** ((**freq** s_cc)⇒(**sometime** (r_cc⊕s_dr)))
：受信側が何度も CC を返せば，いつかは送信側に CC が受け取られる．または，送信側が DR を送る．

(2′) データ転送フェーズ

(2′.1) **ALWAYS** (r_cc⇒(**SOMETIME** (s_dt⊕s_dr)))
：送信側が CC を受け取れば，送信側はデータを送る．または，送信側が DR を送る．

(2′.2) **ALWAYS** ((**freq** s_dt)⇒(**sometime** (r_dt⊕s_dr)))
：送信側が何度もデータを送れば，いつかは受信側にデータが受け取られる．または，送信側が DR を送る．

(2′.3) **ALWAYS** (r_dt⇒(**SOMETIME** (s_ak⊕s_dr)))
：受信側がデータを受け取れば，受信側は AK を返す．または，送信側が DR を送る．

(2′.4) **ALWAYS** ((**freq**s_ak)⇒(**sometime**(r_ak⊕s_dr)))

```
process Proc1[s_cr, r_cc, s_dr, r_dc] :=
  s_cr ; ( r_cc ; s_dr ; Proc2[s_cr, r_cc, s_dr, r_dc]
         [] s_dr ; Proc2[s_cr, r_cc, s_dr, r_dc] )
where
  process Proc2[s_cr, r_cc, s_dr, r_dc] :=
      r_dc ; Proc1[s_cr, r_cc, s_dr, r_dc]
  endproc
endproc
```

図 13 アブラカダブラプロトコルの全体的な時間的性質から得られた LOTOS 仕様
Fig. 13 The LOTOS specification of Abracadabra protocol obtained from the global properties.

```
process Proc1[Δ] :=
  s_cr ; Proc2[Δ]
where
  process Proc2[Δ] :=
    hide int1 in
      int1 ; s_cr ; Proc2[Δ]
    [] r_cr ;
        ( s_cc ; Proc3[Δ]
        [] s_dr ; Proc7[Δ] )
    [] s_dr ; Proc7[Δ]
  endproc
  process Proc3[Δ] :=
    hide int2 in
      int2 ; s_cc ; Proc3[Δ]
    [] r_cc ; Proc4[Δ]
    [] s_dr ; Proc7[Δ]
  endproc
  process Proc4[Δ] :=
      s_dt ; Proc5[Δ]
    [] s_dr ; Proc7[Δ]
  endproc
  process Proc5[Δ] :=
    hide int3 in
      int3 ; s_dt ; Proc5[Δ]
    [] r_dt ;
        ( s_ak ; Proc6[Δ]
        [] s_dr ; Proc7[Δ] )
    [] s_dr ; Proc7[Δ]
  endproc
  process Proc6[Δ] :=
    hide int4 in
      int4 ; s_ak ; Proc6[Δ]
    [] r_ak ; Proc4[Δ]
    [] s_dr ; Proc7[Δ]
  endproc
  process Proc7[Δ] :=
    hide int5 in
      int5 ; Proc8[Δ]
    [] r_dr ; s_dc ; Proc8[Δ]
  endproc
  process Proc8[Δ] :=
    r_dc ; Proc1[Δ]
  endproc
endproc
```

図 14 より詳細なアブラカダブラプロトコルの LOTOS 仕様
Fig. 14 The more detailed LOTOS specification of Abracadabra protocol.

: 受信側が何度も AK を返せば，いつかは送信側に AK が受け取られるか，または送信側が DR を送る．

(2′.5) **freq** s_dt

: データの転送は何度でも可能である．

(3′) コネクション切断フェーズ

(3′.1) **ALWAYS** (s_dr⇒(**SOMETIME** (r_dr⊕(**SOMETIME** r_dc))))

: 送信側が DR を送れば，受信側が DR を受け取るか，または，送信側が DC を受け取る．

(3′.2) **ALWAYS** (r_dr⇒(**SOMETIME** s_dc))

: 受信側が DR を受け取れば，受信側は DC を返す．

(3′.3) **ALWAYS** (s_dc⇒(**SOMETIME** r_dc))

: 受信側が DC を返せば，送信側は DC を受け取る．

6.2 時間的性質を満足するプロトコル仕様

詳細は省略するが，上の性質から構成法1に従って，プロトコル全体の振舞いを表す LOTOS 仕様が，また，2に従って，より詳細な LOTOS 仕様が得られる．得られた仕様を**図 13, 14** に示す．ただし，図 14 において，Δ は，

s_cr, r_cr, s_cc, r_cc, s_dt, r_dt, s_ak, r_ak, s_dr, r_dr, s_dc, r_dc

である．

7. 結　び

本論文では，LTS から得られるアクション系列集合をモデルとする拡張分岐時制論理を定義し，時間的性質を満足する LTS を合成する方法を示した．さらに，プロトコルの時間的性質として生存性を用いて LOTOS 仕様を導出することができることを示した．また，時間的性質を段階的に与えることにより，段階的な仕様化を行うこともできる．一般に，時間的性質を満足する LTS は一通りには決定できないが，本論文ではアクション数が限定された LTS を選び，仕様の時間的性質の充足に関しての標準を与えた．

この構成法では原子的な LTS を用いており，この構成法のみを用いて大規模な仕様を構成することはそれほど容易なことではない．しかしながら，大局的な性質から仕様を得ることができることから，仕様化の初期段階で本構成法を適用することは有効である．

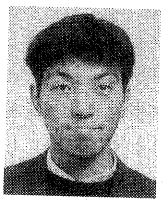
今後の展望としては，並列性を考慮した仕様の構成法の導出，および，本構成法の支援系の実現を予定している．それによって，時制論理に基づいた大規模な仕様化が容易となることが期待できる．

参 考 文 献

1) ISO/IEC : Information Technology—Open Systems Interconnection—Guidelines for the Application of Estelle, LOTOS and SDL, ISO/IEC TR 10167 (1991).
2) Gotzhein, R.: Temporal Logic and Applications—A Tutorial, *Computer Networks and*

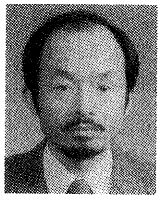
ISDN System, Vol. 24, pp. 203-218, North-Holland (1992).
3) Ben-Ari, M., Manna, Z. and Pnueli, A.: The Temporal Logic of Branching Time, *Proc. of 8th Annual ACM Symposium on Principles of Programming Languages*, pp. 164-176 (1981).
4) Cavalli, A. R. and Horn, F.: Proof of Specification Properties By Using Finite State Machines and Temporal Logic, *Protocol Specification, Testing, and Verification VII*, pp. 221-233, Elsevier Science Publishers B. V., North-Holland (1987).
5) Fantechi, A., Gnesi, S. and Laneve, C.: An Expressive Temporal Logic for Basic LOTOS, *Formal Description Techniques II*, pp. 261-276, Elsevier Science Publishers B. V., North-Holland (1990).
6) ISO: Information Processing Systems—Open Systems Interconnection—LOTOS—A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, ISO 8807 (1989).
7) 安藤敏彦, 加藤 靖, 高橋 薫, 野口正一：時制論理に基づくプロトコルの LOTOS 仕様の合成, 情報処理学会研究報告, 92-SE-87 (1992).

(平成4年10月26日受付)
(平成5年 4月 8日採録)



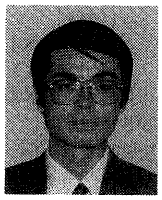
安藤 敏彦（正会員）

1963 年生．1987 年東北大学工学部原子核工学科卒業．1989 年同大学院博士前期課程修了．同年仙台電波工業高等専門学校情報工学科助手，現在に至る．仕様記述，セル構造オートマトンの研究に従事．物理学会，形の科学会各会員．



加藤 靖（正会員）

1978 年東北大学大学院博士課程修了．工学博士．現在国立仙台電波工業高等専門学校情報工学科助教授．セル構造オートマトンの動特性，ディジタル技術の教育方法，形式仕様記述言語および時制論理によるプロトコルの設計・検証に関する研究に従事．著書「情報数学」（共立出版），「基本的プログラミングと割込み」，「マイクロコンピュータ周辺・割込み技術」（共著，日刊工業新聞社）．電子情報通信学会，日本工業教育協会各会員．



高橋 薫（正会員）

1954 年生．東北大学電気通信研究所助手，工学博士．分散システム・通信プロトコルの設計法，仕様記述技法，検証・論理検証法，合成法，試験法，通信ソフトウェアの自動インプリメンテーション法などの研究に従事．電子情報通信学会，IEEE Computer Society 各会員．現在，(株)高度通信システム研究所客員研究員．

野口 正一（正会員）

昭和5年生．昭和 29 年東北大学工学部電気工学科卒業．昭和 35 年同大学院博士課程修了．工学博士．昭和 46 年東北大学電気通信研究所教授．昭和 59 年東北大学大型計算機センター長．平成2年東北大学応用情報学研究センター長．主として情報システム構成論，知識処理に関する研究に従事．著書「情報ネットワーク理論」（岩波書店），「知識工学基礎論」（オーム社）など．現在，日本大学工学部教授．