東北大学審査博士学位論文

通信システムの適合性試験に関する研究

電気及通信工学専攻

提出者 岡崎 直宣

東北大学審査博士学位論文

通信システムの適合性試験に関する研究

提出者 岡崎 直宣

論 文 題 目 通信システムの適合性試験に関する研究

提出年月日 平成 4 年 1 月 20 日

博士学位論文公表目録

公表年月日	論文題目	発表雑誌及発行所名	備考
平成3年10月25日	LOTOS仕様からの効率的な試験系列の生成法	電子情報通信学会論文誌B-I	第4章に該当
投稿中	通信システムにおける相互接続試験系列生成法	電子情報通信学会論文誌B-I	第3章に該当

※ 備考欄には、公表論文が本論文のどの部分に該当するかを記入すること。

指導教官 東北大学教授

野口 正一

平成3年度 博士学位論文

通信システムの適合性試験に関する研究

岡崎 直宣

目　次

第1章　序論

1.1 背景と目的

近年, 情報通信システムなどの分散処理システムが急速に普及拡大してきている. これに伴い, 特に, 大規模化, 複雑化, 多様化する傾向にある通信ソフトウェアをいかに効率的に開発するかが, 非常に重要な研究課題となっている. このような状況において, 開発された製品に対する試験の重要性がますます高まっている.

現在, 通信プロトコル等通信システムを対象とした製品試験(広義の適合性試験, 以下ではこれを単に試験と呼ぶ)は, 個々の製品に対して個別に行う“狭義の適合性試験(以下ではこれを適合性試験と呼ぶ)”と, 実利用環境のもとで実際に製品同士を接続して行う“相互接続試験”との組合せで実施されている. このうち, 適合性試験については, その方法と枠組みの標準化がISOで進められている[ISO89c].

試験に関する問題の中で主要な部分を占めるものの1つが, 試験系列の生成に関する問題である. これは, 現状では人手で行われており, 試験のコストや信頼性の面で問題があった.

通信システムの記述は, 従来は自然言語等非形式的な方法によって行われていたが, その解釈の曖昧さから生じる問題があった. そこで, FDT(Formal Description Technique)と呼ばれる曖昧性のない形式的な言語がISO等によって開発され, 種々の通信プロトコルがこのFDTにより記述されるように

なった．今後はさらに多くの通信プロトコル等がFDTにより記述されていくものと予想される．

現在，国際機関によって標準として定められているFDTは，FSM(Finite State Machine)をその基礎とするもの[CCIT 89] [ISO 89b]と，イベントの時間順序に基づくもの[ISO 89a]がある．試験系列の生成に関する研究としては，適合性試験について，前者の基礎となるFSMより試験系列の生成を行う方法がいくつか発表されている[Nait 81] [Chow 78] [Gone 70] [Sabn 88] [Sato 91]．また，イベントの時間順序に基づくFDTより適合性試験系列の生成を行う非形式的手法も発表されている[Guer 89]．一方，相互接続試験については，試験系列の生成法に関しての研究はほとんどなされていないのが現状である．

本論文では，通信システムの試験における試験系列の生成法に関して，2つのアプローチにより研究を行う．

一つは，FSMを対象とした相互接続試験系列の生成に関する研究である．これは，従来のFSMより適合性試験系列の生成を行う方法[Nait 81] [Chow 78] [Gone 70] [Sabn 88] [Sato 91]を発展させ，相互接続試験においても試験系列の生成ができるようにするものである．

一般に，N個のエンティティが相互接続された環境において，個々のエンティティの動作を規定する仕様より従来の適合性試験に関する試験系列の生成法を用いてN個の試験系列を個別に得ることができる．しかし，相互接続試

験においては，N個の試験系列中に含まれるアクション間の相互の時間的前後関係が非常に重要であり，このN個の試験系列だけでは試験を行うことができない．そこで，相互接続試験においては，このような従来の適合性試験に関する試験系列の生成法とは異なる手法が必要である．

本論文では，上のNがN＝2の場合について，相互接続試験における試験系列の生成手法を提案する．本手法では，双方向のチャネルにより相互接続された2つのエンティティの動作を規定するプロトコル仕様を対象とする．ここでは，仕様は有限状態機械(FSM:Finite State Machine)により記述されるものとする．そして2つのエンティティとその間のチャネルを合成したものを一つのシステムと見なし，そのシステム全体の振る舞いを調べることを相互接続試験と考える．また試験環境としては，2つの試験対象に対してそれぞれ上位レイヤ，下位レイヤの位置にテスタ(それぞれ上位テスタ，下位テスタと呼ぶ)を置き，制御，観測する環境を想定する．本手法で得られる試験系列は，2つの試験対象の状態とその間のチャネルの状態を合成したシステム状態の遷移を調べるための，2つの試験対象とそれぞれの上位テスタ及び下位テスタとの間の入出力の系列である．

本手法では，システム全体の振る舞いを示すシステム状態グラフを導入する．そして，2つのFSMにより記述されたプロトコル仕様よりシステム状態グラフを生成する手順をルールとして与える．さらに，ここではチャネル

にエラーがある場合を想定したプロトコル仕様に対しても適用できるようこの生成ルールを拡張する．この拡張されたルールによって得られる拡張システム状態グラフは，相互接続試験時に動作すべきシステム全体の振る舞いを表しており，ここには2つのエンティティのアクション間の相互の時間的前後関係が明確に表されている．最後に，この拡張システム状態グラフに対し従来の適合性試験に対する試験系列生成法を適用することによって，相互接続試験系列を生成する．本手法を用いることによって，相互接続試験における試験系列生成のコストの削減，試験の信頼性の向上などが期待される．

通信システムの試験における試験系列の生成法に関する二つめのアプローチは，イベントの時間順序に基づくFDTを対象とした適合性試験系列の生成に関する研究である．これは，従来ある非形式的手法[Guer 89]と異なり，弱bisimulation等価と呼ばれる等価性に基づき試験系列の生成を行う形式的手法である．

従来イベントの時間順序に基づくFDTを対象とした適合性試験系列の生成に関する研究として，LOTOSを対象とした試験系列生成法[Guer 89] [Tret 89]がある．このうち，[Tret 89]は仕様からテスタの仕様を導出する手法である．そして，テスタの仕様に基づいて(正しく)実現されたテスタプロセスと試験対象のプロセス(IUT : Implementation Under Test)とを，LOTOSのオペレータ“||”で定義されるような同期的な並列動作を実現する環境の下

で動作させた結果, LOTOSの"stop"に相当するようなデッドロック状態に陥るかどうかで適合性を調べようとするものである. 但し, ここでは有限動作に限定していることによるその適用範囲に関する限界とともに, 上記のような試験環境をどのように実現するか等, 未解決な点が多い. また, [Guer 89]はLOTOSインタプリタを用いて実行木を作り, それよりTTCN表現の試験ケースを生成する非形式的な方法である.

本論文では, LOTOS仕様からの試験系列の自動生成の新たな方法として, 弱bisimulation等価[ISO 89a]の概念を用いて木状の試験系列を生成する形式的な手法を提案する. ここでは, 同期通信によりIUTと通信を行なうテスタを想定し, そのテスタとIUTとの間で行なわれるべき通信の時間的順序を表したものを試験系列(但し, 各種の試験アーキテクチャには独立な抽象試験系列)と考える. 本手法では, データ構造を含むFull-LOTOSを対象とし, また[Tret 89]では扱えないような無限動作が含まれる場合もここでは扱う. 特に, LOTOSの意味表現であるラベル付き遷移システム(LTS : Labelled Transition System[ISO 89a])に変数表現を加えた変数付きLTS(VTS : Valued Transition System)を導入することにより, 試験系列に変数表現を取り入れることができる. これによって, 表現がコンパクトになるだけでなく, 試験実施時における系列の選択に有効な情報を試験系列に与えることができる. さらに, 弱bisimulation関係によるLTSの等価性の概念[Kami 90]を拡張し,

VTS上での状態の等価性を定める．そして，この等価性に基づくVTSの簡約化の方法を与え，この簡約化を繰り返すことによりVTSを既約形に変換する．この変換によって，より効率的な試験系列を生成することができる．試験系列は，木状のLOTOS表現であるTreeLOTOSによって表される．

1.2 論文の構成

本論文の構成は以下の通りである．まず，**第2章**では，通信システムの試験について，適合性試験と相互接続試験の関係，試験の方法論，試験系列の生成に関する従来の研究について述べる．**第3章**では，**FSM**を対象とした相互接続試験系列の生成法を与える．**第4章**では，イベントの時間順序に基づく**FDT**を対象とした形式的な適合性試験系列の生成法を与える．**第5章**は本論文の結論である．

第2章　通信システムにおける試験

2.1 はじめに

本章では，準備として，本論文での議論に必要になる通信システムの試験に関するまとめを行う．

まず，2.2で適合性試験と相互接続試験の関係について述べ，2.3で試験の方法論について紹介する．2.4では通信システムのモデルと通信の形態について述べ，それに基づいて2.5及び2.6でモデルごとの試験系列の生成に関する概念や基本的手法について述べる．2.7は本章のまとめである．

2.2 適合性試験と相互接続試験

本節では，適合性試験と相互接続試験の関係について述べる．

適合性試験(Conformance Testing)は，仕様に基づいて開発された製品が，仕様の規定に適合して動作するかどうかを試験するものである．従って，適合性試験は製品間の相互接続性を直接保証するものではなく，製品間の相互運用を実現するまでの一ステップであり，仕様に適合していることを試験することにより製品相互間の相互接続性を向上させることを目的としている．製品間の相互運用を実現するまでの次のステップとして，相互接続試験(Interoperability Testing)がある．これは，適合性試験に合格した製品同士が，実際の利用環境化で相互接続されて規定通り機能するかどうかを調べるものである．

2.3 試験方法論

本節では，試験の方法論について紹介する．

試験の対象はIUT(Implementation Under Test)と呼ばれ，被試験システム(SUT : System Under Test)の内部におかれる．IUTは通信プロトコルの層モデルの単一の層(例えばOSIモデルのトランスポート層)であることも，複数の層(例えばOSIのセッション層からアプリケーション層まで)のひとまとまりの場合もある．テスタはIUTと交信して試験を進めていく．この試験の内容である試験項目の集まりを試験系列あるいはテストスイートと呼ぶ．

次に，文献[ISO89c]に基づいて，適合性試験に関する試験法について紹介する．図2-1に，試験対象を階層モデルにもとづく開放型システム上に実装された第(N)層((N)エンティティ)ととらえた場合の概念図を示す．このとき適合性試験とは，(N)エンティティのふるまいが規格に適合しているかどうかを調べることである．上位層あるいは下位層とのやりとりは“ASP”(Abstract Service Primitive)と呼ばれる．また，試験対象のふるまいを調べる場所は，“PCO”(Points of Control and Observation)と呼ばれる．

試験対象のふるまいをどのように調べるか，すなわちASPをどのような方法で制御･観測するかにより，いくつかの試験法が提案されている．

試験法としては，現在ローカル試験法，分散試験法，調和試験法，遠隔試験

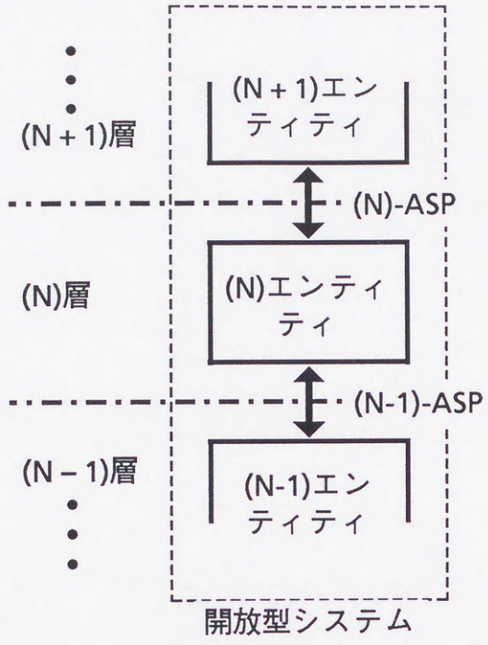
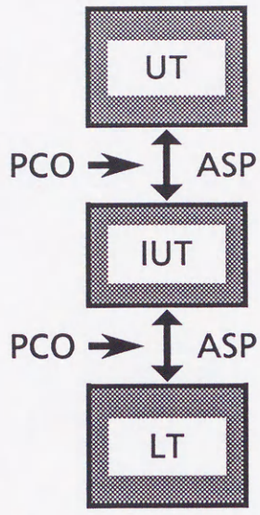


図2-1　階層モデルにもとづく開放型システム

法等が考えられている.

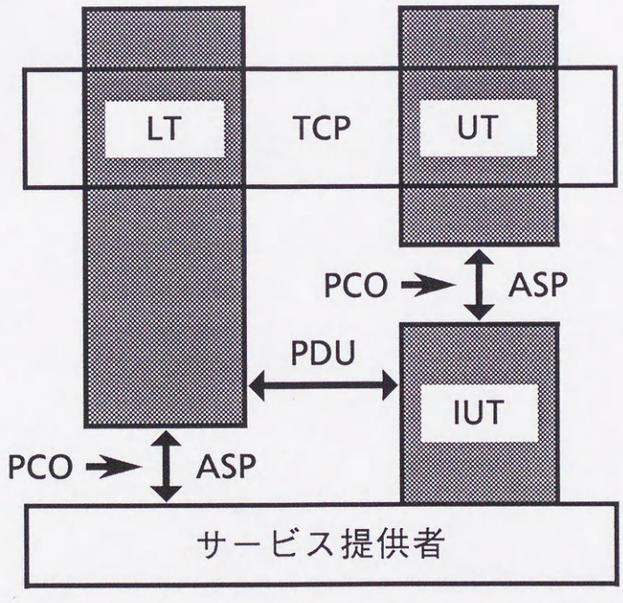
このうち, ローカル試験法(図2-2)は, テスタをSUT内部のみに配置し, IUTの上部の層との境界及び下位の層との境界にPCOを設ける方法である. 被試験システム内に観測点を設けるので詳細な試験データを収集することができるというメリットがある.

また, 分散試験法(図2-3)は, SUTの内部及び遠隔システム内にテスタを配置し, IUTの上部の層との境界及び遠隔システム内にPCOを設ける方法である. 上位テスタ及び下位テスタのASP操作は, 試験調和手順(TCP : Test Coordination Procedure)に従って行われる. この方法も, 被試験システム内に観測点を設けるという点でローカル試験法と同様のメリットがある.



IUT : Implementation Under Test　ASP : Abstract Service Primitive
UT : Upper Tester　PCO: Point of Control and Observation
LT : Lower Tester

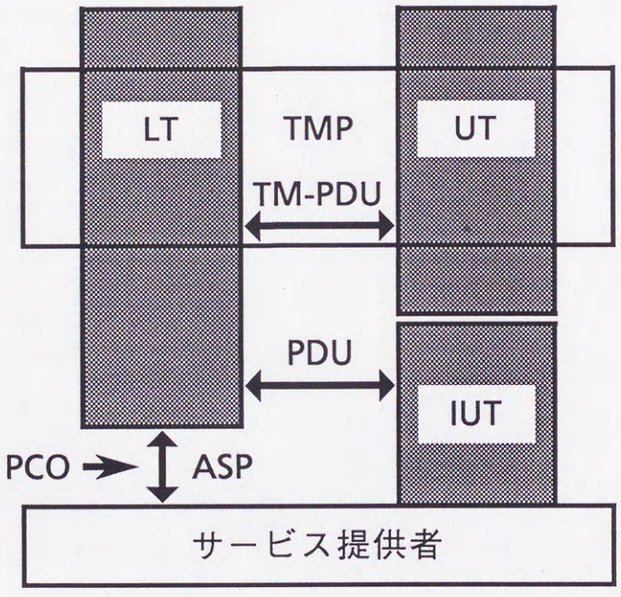
図2-2　ローカル試験法



IUT : Implementation Under Test　ASP : Abstract Service Primitive
UT: Upper Tester　PCO: Point of Control and Observation
LT: Lower Tester　TCP : Test Coordination Procedure
PDU: Protocol Data Unit

図2-3　分散試験法

調和試験法(図2-4)は，SUTの内部及び遠隔システム内にテスタを配置する．そして，上位テスタと下位テスタとが強調して試験を実行する試験管理プロトコル(TMP: Test Management Protocol)が定義されており，制御及び観測は試験管理PDU(TM-PDU : Test Management-PDU)の制御と観測を含め下位テスタのみで実施することができる．

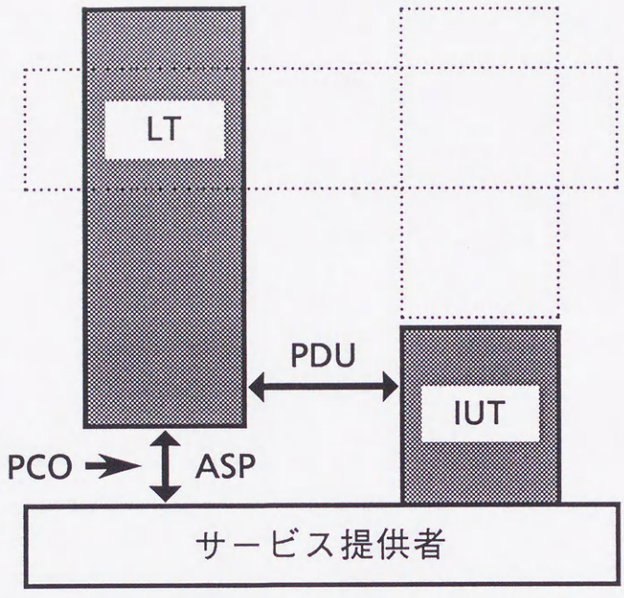


IUT : Implementation Under Test　ASP : Abstract Service Primitive
UT : Upper Tester　PCO: Point of Control and Observation
LT : Lower Tester　TMP : Test Management Protocol
PDU : Protocol Data Unit　TM-PDU : Test Management PDU

図2-4　調和試験法

遠隔試験法(図2-5)は，遠隔システム内のみにテスタを配置し，IUTの上部の観測点や試験管理プロトコルはない．被試験システム上に上位テスタを設ける手間を省くことができる．

現在，このような方法に基づく適合性試験が，ヨーロッパ，アメリカで実



IUT : Implementation Under Test　　ASP : Abstract Service Primitive
LT : Lower Tester　　PDU : Protocol Data Unit
PCO: Point of Control and Observation

図2-5　遠隔試験法

施され，日本ではINTAP(財団法人情報処理相互運用技術協会)の試験センターが平成元年3月から適合性試験サービスを開始している．

本論文の第4章で提案するLOTOS仕様からの効率的な試験系列の生成法においては，特定の試験法に依存しない抽象試験系列の生成を行う．従って，この手法で得られる試験系列は基本的に上のどの試験法にも適用可能である．特に，ローカル試験法にはそのまま適用可能である．また，分散試験法及び調和試験法においてはTCPあるいはTMP等のための補助的な情報を補うことによって，さらに遠隔試験法においては上位テスタに関する情報を無視することによって，それぞれ適用可能である．

次に，相互接続試験に関する試験法について述べる．ここでは，特に本論文の第3章で提案する通信システムにおける相互接続試験系列生成法における試験対象システムの環境のモデル及び相互接続試験のアーキテクチャについて述べる．

本手法では，図2-6のような相互接続試験における被試験システムの環境をモデルとする．すなわち，被試験システムとしては，第N層の通信し合う任意のプロトコルエンティティ，N-Entity1およびN-Entity2を考え，N-Entity1, N-Entity2とそれぞれ(N)SAP (Service Access Point)を通して接続される上位層をそれぞれユーザ1, ユーザ2と呼ぶ．また，タイマー処理を扱う特別なAP(Access Point) Tを通してN-Entity1, N-Entity2と接続されるタイマープロセスをTM1, TM2と呼ぶ．さらに，N-Entity1およびN-Entity2はそれぞれ(N-1)SAPを通して双方向のFIFOチャネルに接続されている．但し，ここではこのチャネルが不完全な場合をも考慮する．具体的には，このチャネルは原則的には入力したデータを反対側に入力順に出力するが，任意のデータが紛失する可能性があるものとする．

また，本手法で対象とする相互接続試験の環境のモデルは図2-7のようになる．被試験システムである2つのプロトコルエンティティN-Entity1およびN-Entity2をIUT(Implementation Under Test)1, IUT2と呼び，それぞれ上位，下位のPCO(Points of Control and Observation)を通して上位テスタ，下

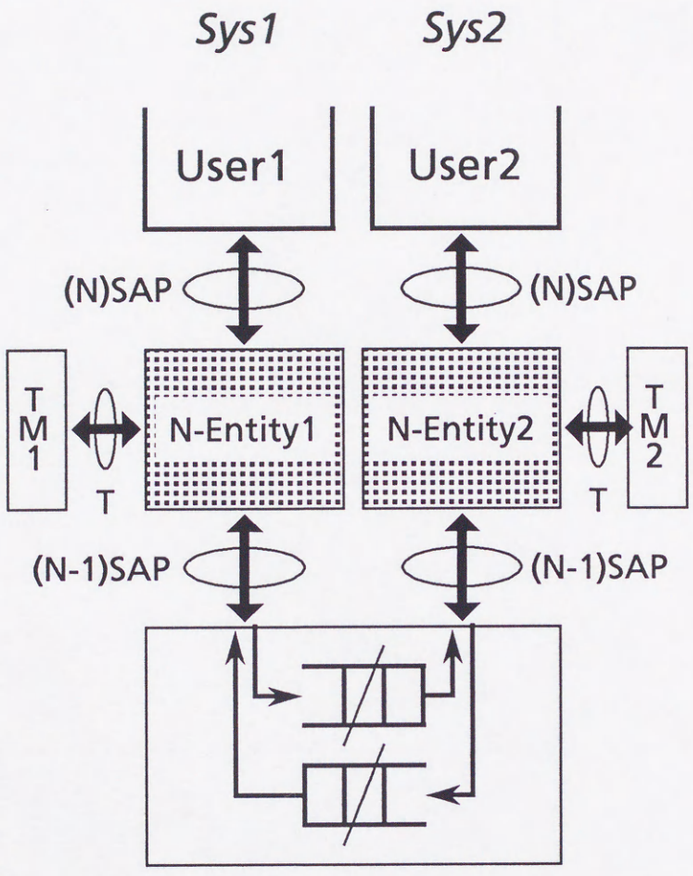


図2-6　被試験システムの環境のモデル

位テスタとデータのやり取りをする．また，2つの下位テスタは，(N-1)層以下のサービス提供者を通して互いに通信する．ただし，ここで受ける(N-1)サービスは完全なものを仮定する．以降，IUTkの上位テスタをUTk，下位テスタをLTk，またUTkとIUTkの間のPCOをUk，LTkとIUTkの間のPCOをLkとそれぞれ呼ぶ．また，IUTkのタイマー処理を扱うPCOをTkと呼ぶ(k＝1,2).

このように，本手法では下位サービスを双方向の「不完全な」FIFOチャ

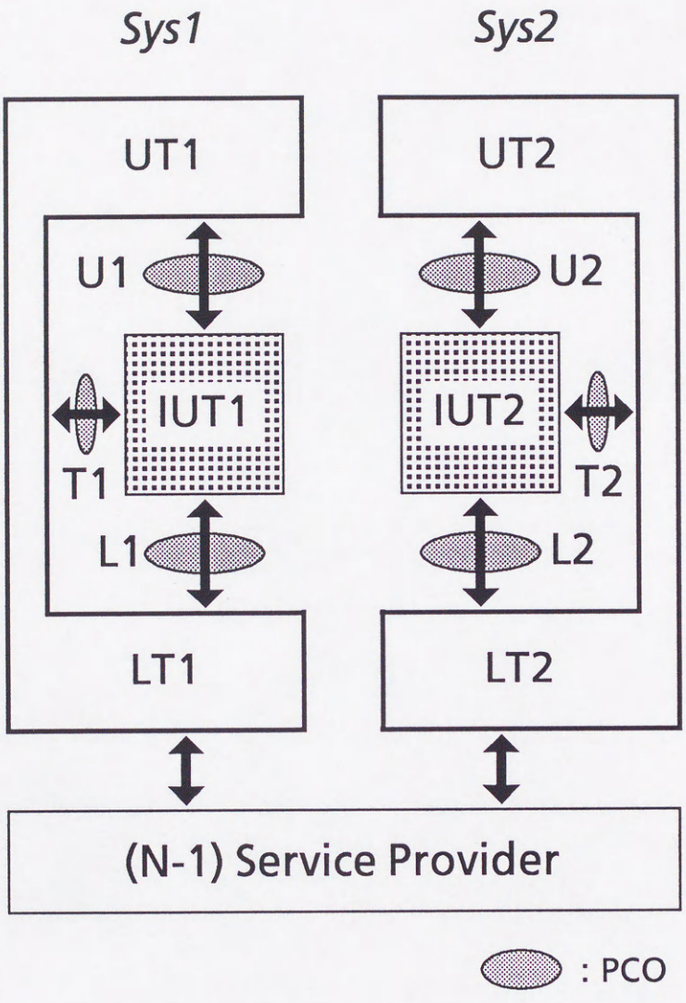


図2-7　相互接続性試験環境のモデル

ネルとしてモデル化した環境を想定したプロトコル仕様を対象とする．一方，その不完全さは試験系列に反映されるため，試験をする際の環境としては「完全な」下位サービスを想定する．

本手法で対象とする相互接続試験の試験系列とは，6つのPCO，U1，U2，L1，L2，T1，T2においてテスタがどの順番でどういったデータをやり取りするかを記述したものであるものとする．なお，2つの下位テスタLT1，LT2

間の通信に関して，および各テスタ間の同期の実現方法等に関しては，別にこれを定めるものとする．

2.4 通信システムのモデルと試験系列生成法

本節では，通信システムのモデルと通信の形態について述べる．

第1章でも述べたように，通信システムのモデルは，FSM(Finite State Machine)モデルとイベントの時間順序に基づくモデルの2つがある．

このうち，FSMモデルはシステムの内部状態とその状態間の遷移によりシステムを記述するものである．このモデルでは，システムの内部状態を陽に記述し，ある状態から入力を受けると，出力を出して別の状態に遷移するといった記述が基本である(図2-8)．

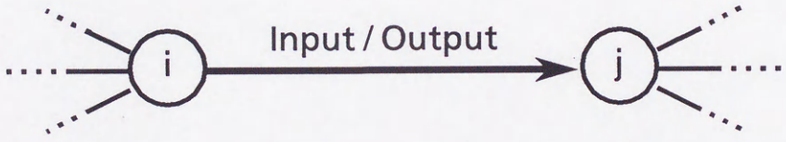


図2-8　FSMモデル

一方，イベントの時間順序に基づくモデルはシステムの外部あるいは環境から見たイベントの時間順序によりシステムを記述するものである．このモデルでは，システムの内部状態を陽に記述せず，外部から観測できるイベントの起こる順序を記述することが基本である(図2-9)．

また，通信システムのモデルの2つのタイプとは別に，システム間の通信の形態には非同期通信と同期通信の2つのタイプがある．

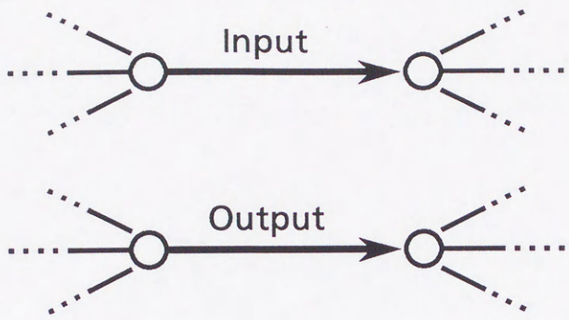


図2-9　イベントの時間順序に基づくモデル

このうち，非同期通信は送信動作と受信動作が独立に起こるような通信の形態である．送信側と受信側の間にバッファを設け，受信側の状態に係わりなく送信側はバッファに対していつでも送信動作をすることが可能であり，逆に受信側は送信側の状態には係わりなくバッファにある信号をいつでも受信することが可能である(図2-10)．この通信の形態においては，送信と受信の区別が重要である．

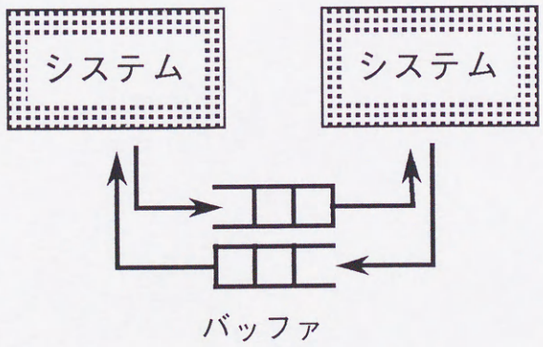


図2-10　非同期通信

一方，同期通信は送信動作と受信動作が常に同時に起こるような通信の形態である．すなわち，受信側が受信動作をできるときのみ送信側の送信動作

が可能である(図2-11). この通信の形態においては, 送信と受信が常に同時に起こるため, その区別は重要ではなく, 何が起こったかだけが重要である. そこで, これらをアクション, あるいはイベントとして抽象化する.

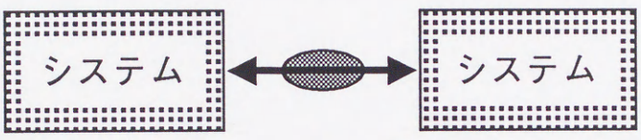


図2-11　同期通信

本来, これらの通信システムのモデルとシステム間の通信の形態は独立に考えうるものであるが, 従来FSMモデルは非同期通信と, またイベントの時間順序に基づくモデルは同期通信とそれぞれ組み合わされて考えられてきた. 本論文でも, 以下ではFSMモデルにおいては非同期通信を, またイベントの時間順序に基づくモデルにおいては同期通信を前提として議論を行う.

以下2.5および2.6では, それぞれのモデルや通信の形態に関する試験系列の性質と試験系列生成法について述べる.

2.5 FSMモデルに関する試験系列生成法

本節では，FSMモデルに関する試験系列生成の概念や基本的手法について述べる．

FSMモデルに関する試験系列生成法については，適合性試験について,従来TT法[Nait 81], PW法[Chow 78], DS法[Gone 70], UI法[Sabn 88]等幾つかの試験系列生成法が提案されている．これらは，いづれも次の定義で定められるようなMealy型FSMをモデルとする．

【 定義2-1 】Mealy型FSM

Mealy型FSM M_M は4項組$<Q, I, O, \omega, \delta>$である．ここで，

Q：状態の集合

I：入力アクションの集合

O：出力アクションの集合

ω：出力関数　　$Q \times I \rightarrow O^*$,　O^*は出力系列の集合

δ：遷移関数　　$Q \times I \rightarrow Q$　　□

Mealy型FSMの特徴である出力関数と遷移関数のうち，TT法においては出力関数のみを，またPW法, DS法, UI法においては出力関数と遷移関数の両方を確認することを目的としている．これらの手法の特徴をまとめると，表2-1のようになる．

手法	制限	調べることができる範囲
TT法	完全定義, 最小, 強連結	出力関数
PW法	完全定義, 最小, 強連結	出力関数, 遷移関数
DS法	完全定義, 最小, 強連結, 判定系列の存在	出力関数, 遷移関数
UI法	完全定義, 最小, 強連結	出力関数, 遷移関数

表2-1　各手法の特徴

これらの試験法は, システムの記述について同表のような制限をし, さらに実現されたシステムにおいては内部状態数が既知であるとの仮定をおいて,

(1) 状態の確認を行う部分

(2) 遷移の確認を行う部分

(3) 遷移先の状態の確認を行う部分

から構成される試験系列を生成する手法である(ただし, TT法においては(2)のみ)[Sidh 89].

本論文の第3章で与えるFSMを対象とした相互接続試験系列の生成法においては, Mealy型FSMを対象とし, 次のように試験を定める.

【 定義2-2 】FSMをモデルとする通信システムの試験

FSMをモデルとする通信システムの試験を次の(1)~(3)を行うことであ

ると定める.

(1) IUTの内部状態の確認

(2) 規定された全ての遷移の確認

(3) 遷移先の状態の確認 □

2.6 イベントの時間順序に基づくモデルに関する試験系列生成法

本節では，イベントの時間順序に基づくモデルに関する試験系列生成の概念について述べる．

2.4で述べたように，イベントの時間順序に基づくモデルは同期通信を前提とする．同期通信システムにおいては，通信はそれに関与するすべてのシステムで同時に起こるため，入出力の区別は重要でなく，“何が”起こったかだけが重要である．そこで，通信の入出力の区別をせずに，それらをイベントあるいはアクションとして抽象化してシステムの動作を表すことができる．

また，仕様記述においてより抽象的な記述を可能とするように，遷移に非決定性を含めることが考えられる．ここで，遷移の非決定性とは，状態とアクションが決まったとき，それによって起こる遷移が1つに定まらないことを言う．さらに，外部環境で制御，観測できないような特別なアクションである内部アクションを取り入れることで，システムの非決定性を明確に表した記述が可能になる．この時，システムの動作の仕様は，次のように定義されるLTSとして表される．

【 定義2-3 】LTS (labelled transition system)[ISO 89a]

LTS Sys は4項組$<S, A, T, s_0>$である．ここで，

S : 状態の集合

A : アクションの集合(内部アクションを含む)

T : 遷移の集合 $T \subseteq S \times A \times S$

s_0 ($\in S$) : Sys の初期状態. □

遷移(s_1, a, s_2)がTの要素であるとき，これはアクション"a"の生起によって状態"s_1"から"s_2"への遷移が起こることを意味し，"$s_1 \text{—}a\rightarrow s_2$"と表す．以下本節ではLTSに基づいたシステムのモデルについて考察を行う．

試験系列生成手法の実用性は，一般に次の2点から論じることができる．

(1) 仕様や実現に対する制限

(2) 得られる試験系列の長さ

以下では，主に(1)の観点から実用性を議論する．

従来のFSMをモデルとする非同期通信システムに関する研究では，システムの記述と実現されたシステムとに次のような仮定をおいていた．

- システムの仕様記述：完全定義，決定的である．
- 実現されたシステム：内部状態数が既知である．

さらに，実現されたシステムに対し，各状態毎にステータスメッセージ(status message)と呼ばれる状態遷移を伴わない状態確認のための入力を仮

定した上で，より短い試験系列を生成する研究もある[Uyar 86]．これらは，いづれも上記のような制限をおくことによって，システムの状態及び遷移を完全に調べることができたが，一方，これらの制限を緩めたより実用性の高い手法が望まれている．

本論文の**第3章**で与えるLOTOS仕様からの効率的な試験系列の生成法においては，同期通信システムにおいて，上のような制限をおかない場合を考える．

そこで，この立場での試験を次のように定める．

【 定義2-4 】LTSをモデルとする同期通信システムの試験

LTSをモデルとする同期通信システムの試験を次の(1)と(2)で定める．任意の時点tにおいて，

(1) IUTと環境との間で取り得るアクションの集合$A_t \subseteq A$(但し，Aはアクションの集合)を調べる．

(2) テスタが任意のアクション$a \in A_t$の生起の環境を提供したときに，IUTがaを行うことを確かめる． □

このように，従来の研究におけるような強い制限をおかない場合には，試験対象の各時点での内部状態の確認ができない．そのため，各回の試験ごとに試験系列の最初から最後までを行う必要がある．このとき，試験系列における合流は意味を持たない．ここで合流とは，一度分岐した2つ以上の系列が

再び1つになることを言う．そこで，試験系列は合流を含まない表現，すなわち木構造の表現として表すことができる．

ところが，仕様には一般に繰り返しであらわされる無限動作が規定されている．この場合，試験系列は無限の木になる．一方，実際の試験において無限の長さの試験は実行不能であるため，このような無限の木の一部のみが実行されるにすぎない．このとき，どこまでの範囲を実行するかは，試験実施者の選択によるべきである(この選択を垂直選択(vertical selection)[Tret 89]と呼ぶ)．

そこで，試験系列としては，無限の動作の意味を表しつつ，垂直選択の目安になる情報を含んだ有限の表現であることが望ましい．

そのために，試験系列の表現方法として，単純な木構造ではなく，合流の中でも特別の性質を持つループを含んだ構造を用いる．以下では，これを“木状構造”と呼ぶ．ここでループとは，一つの系列においてその系列上に合流するものを言う(詳細な定義については4.3を参照されたい)．このように，木状構造にループを取り入れることで，無限の動作を有限の表現として表すことができ，また，このループを何回繰り返すかが，垂直選択の目安になると考えられる．

以下では，このような性質を持った試験系列を“試験スイート”と呼び，次のように定める．

【 定義2-5 】試験スイート

次の(1)から(3)を満たすような木状構造を試験スイートと定める.

(1) 節点が試験における各時点に対応する.

(2) 枝が各時点においてIUTが取るべきアクションに対応する.

(3) 根から1つの葉へたどるパスには1回の試験でたどるアクションの時間系列(試験ケース)が表されている. □

なお, 仕様中に非決定性が含まれる場合, それを反映して試験スイートに非決定性が含まれる. この試験スイート中の非決定性は, 試験の種類や状況により試験実施者が選択して実行するものとする.

また, 従来の試験系列の能力の定義に代わって, ここでの試験スイートの能力について, 次のように定める.

【 定義2-6 】試験スイートの能力

試験スイートは, それが仕様と等価であるとき, その時に限って, その能力が完全であるという. □

ここで, "等価"とは, 第4章で導入する弱bisimulation等価をさす.

以下第4章では, 仕様記述言語として, 同期通信でLTSを意味モデルとするLOTOSを対象とし, LOTOSで記述された仕様から試験スイートを生成する手法について考察する. 試験スイートは木構造を持つLOTOS表現であるTreeLOTOSによって表す.

2.7 まとめ

本章では，準備として，本論文での議論に必要になる通信システムの試験に関するまとめを行った．

まず，適合性試験と相互接続試験の関係について述べ，それぞれに関する試験の方法論を紹介した．

さらに，通信システムのモデルとして非同期通信を前提としたFSMモデルおよび同期通信を前提としたイベントの時間順序に基づくモデルの2つが考えられていることを述べた．そして，それぞれのモデルに対する，試験系列の生成に関する概念や基本的手法について述べた．

第3章　通信システムにおける相互接続試験系列生成法

3.1 はじめに

本章では，通信システムにおける相互接続試験系列の生成法を与える．

3.2でシステム全体の振る舞いを示すシステム状態グラフを導入する．そして，2つのFSMにより記述されたプロトコル仕様よりシステム状態グラフを生成する手順をルールとして与える．3.3では，チャネルにエラーがある場合を想定したプロトコル仕様に対しても適用できるようこの生成ルールを拡張する．3.4では3.3で得られた拡張システム状態グラフを変形し，従来の適合性試験に対する試験系列生成法が適用可能な形に導き，相互接続試験系列を生成する手法を与える．3.5では以上の手法の適用例を示す．そして最後に3.6で本章のまとめについて述べる．

3.2　システム状態グラフの導入

本手法では，次のように定められる，通信し合う2つのFSMでモデル化されたプロセスから成るプロトコルを対象とする．

【 定義3-1 】プロトコル$P=<P_1,P_2>$

ここで，

P_k：プロセスk ($k=1,2$)

$P_k=<Q_k,A_k,Tr_k,o_k,F_k>$

Q_k：プロセスkの状態の集合

A_k：プロセスkの入出力アクションの集合

$A_k=\{pd \mid p\in SAP_k, d\in D\}\cup\{T_k-start, T_k+timeout\}$

SAP_k：プロセスkのSAP(Service Access Point)の集合

$d\in D$ は $-m_o$ 又は $+m_i$ の形

但し，$m_o\in M_{kj}\cup M_{ko}$, $m_i\in M_{jk}\cup M_{ki}$,

M_{kj}：プロセスkからプロセスjへ送り得るメッセージの集合

($j=1,2,\ k\neq j$)

M_{jk}：プロセスjからプロセスkへ送り得るメッセージの集合

($j=1,2,\ k\neq j$)

M_{ko}：プロセスkからユーザkへの送り得るメッセージの集合

M_{ki}：ユーザkからプロセスkへの送り得るメッセージの集合

T_k：プロセスkのタイマーアクセスポイント

Tr_k：プロセスkの状態遷移を表す関係

$Tr_k \subseteq Q_k \times A_k \times Q_k$

o_k：プロセスkの初期状態

F_k：プロセスkの最終状態の集合　□

$e=p-m \in A_k$のとき，eを入力アクションと呼び，$e=p+m \in A_k$のとき，eを出力アクションと呼ぶ．

プロトコルが上の定義のように2つのFSMとして表されるプロセスから成るとき，各プロセスについて，それぞれに従来の適合性試験の試験系列生成法を適用することにより，2つの試験系列を得ることができる．ところが，相互接続試験においてはこの独立に得られた2つの試験系列中の各アクション間の前後関係が重要である．例えば，2つの試験系列中のあるアクション系列が$a_1b_1c_1$および$d_2e_2f_2$であった時，b_1はa_1の後でc_1の前に起こることは分かるが，b_1とd_2, e_2, f_2との前後関係の情報が欠けているため，この2つの系列からは相互接続試験を行うことができない．この様な問題をここでは相互接続試験における同期の問題と呼ぶ．

そこで，本手法では，この同期の問題を解決するために，従来プロトコルの論理エラーの検出法などに用いられているパータベーションの手法

[West 78]の考え方を発展させ，システム状態グラフ(SSG : System States Graph)を導入する．SSGは，2つのプロセスの状態及びその間のチャネルの状態を合成して得られるシステム状態をノードとするグラフである．以下にこの定義を与える．尚，各チャネルはFIFO(First In First Out)であるものとする．

【 定義3-2 】システム状態

システム状態 s は4項組$<q_1,q_2,c_{12},c_{21}>$である．但し，

q_k : プロセスkの状態　$q_k \in Q_k$

c_{12} : プロセス1からプロセス2へのチャネルの内容

$c_{12} \in M_{12}^*$

c_{21} : プロセス2からプロセス1へのチャネルの内容

$c_{21} \in M_{21}^*$ □

【 定義3-3 】SSG

SSG　G_Sは4項組$<S,A,Tr,s_0>$である．ここで，

S : システム状態の集合

A : 入出力アクションの集合

Tr : 遷移関係　$Tr \subseteq S \times A \times S$

s_0 : 初期システム状態 □

次に，2つのプロセスからなるプロトコルPが与えられたとき，このSSGを生成するルールを以下のruleA～ruleIとして与える．このうち，ruleA及びruleD～ruleGは従来のパータベーションの手法によるものと同じであり，ruleB，ruleCはそれぞれプロセス1，プロセス2とユーザとのやりとりによる遷移を拡張したもの，又，ruleH，ruleIはタイマーアクションによる遷移を拡張したものである．

【 SSG生成規則 】

プロトコル$P=<<Q_1,A_1,Tr_1,o_1,F_1>,<Q_2,A_2,Tr_2,o_2,F_2>>$が与えられたとき，$A=A_1 \cup A_2$と次のrule A～rule Iにより推論される最小の集合S,Tr及び初期システム状態s_0からなるSSG $G_S=<S,A,Tr,s_0>$をPに関するSSGと定める．

以下のruleにおいて，Eはチャネルの内容が空であることを示す．また，append(c_{kj},m)，top(c_{kj})，remain(c_{kj})をそれぞれ次のように定める．

append(c_{kj},m) ： チャネルc_{kj}の最後にmを加えた状態のチャネル．

top(c_{kj}) ： チャネルc_{kj}の先頭の内容．

チャネルc_{kj}が空の場合には定義されない．

remain(c_{kj}) ： チャネルc_{kj}の先頭の内容を取り去った残りの状態のチャネル．チャネルc_{kj}が空の場合には定義されない．

rule A (初期システム状態)

$\vdash s_0 = <o_1, o_2, E, E> \in S$

rule B (U_1での入出力)

$s = <q_1, q_2, c_{12}, c_{21}> \in S,\ (q_1, U_1 d, q_1') \in Tr_1,\ U_1 d \in A_1$

$\vdash\ (s, U_1 d, s') \in Tr,\ s' \in S$

但し, $s' = <q_1', q_2, c_{12}, c_{21}>$

rule C (U_2での入出力)

$s = <q_1, q_2, c_{12}, c_{21}> \in S,\ (q_2, U_2 d, q_2') \in Tr_2,\ U_2 d \in A_2$

$\vdash\ (s, U_2 d, s') \in Tr,\ s' \in S$

但し, $s' = <q_1, q_2', c_{12}, c_{21}>$

rule D (L_1での出力)

$s = <q_1, q_2, c_{12}, c_{21}> \in S,\ (q_1, L_1 - m, q_1') \in Tr_1,\ L_1 - m \in A_1$

$\vdash\ (s, L_1 - m, s') \in Tr,\ s' \in S$

但し, $s' = <q_1', q_2, append(c_{12}, m), c_{21}>$

rule E (L_2での出力)

$s = <q_1, q_2, c_{12}, c_{21}> \in S,\ (q_2, L_2 - m, q_2') \in Tr_2,\ L_2 - m \in A_2$

$\vdash\ (s, L_2 - m, s') \in Tr,\ s' \in S$

但し, $s' = <q_1, q_2', c_{12}, append(c_{21}, m)>$

rule F (L_1での入力)

$s=<q_1,q_2,c_{12},c_{21}>\in S$, $top(c_{21})=m$,

$(q_1,L_1+m,q_1')\in Tr_1$, $L_1+m\in A_1$

$\vdash$ $(s,L_1+m,s')\in Tr$, $s'\in S$

但し, $s'=<q_1',q_2,c_{12},remain(c_{21})>$

rule G (L_2での入力)

$s=<q_1,q_2,c_{12},c_{21}>\in S$, $top(c_{12})=m$,

$(q_2,L_2+m,q_2')\in Tr_2$, $L_2+m\in A_2$

$\vdash$ $(s,L_2+m,s')\in Tr$, $s'\in S$

但し, $s'=<q_1,q_2',remain(c_{12}),c_{21}>$

rule H (T_1での入出力)

$s=<q_1,q_2,c_{12},c_{21}>\in S$, $(q_1,T_1d,q_1')\in Tr_1$, $T_1d\in A_1$

$\vdash$ $(s,T_1d,s')\in Tr$, $s'\in S$

但し, $s'=<q_1',q_2,c_{12},c_{21}>$

rule I (T_2での入出力)

$s=<q_1,q_2,c_{12},c_{21}>\in S$, $(q_2,T_2d,q_2')\in Tr_2$, $T_2d\in A_2$

$\vdash$ $(s,T_2d,s')\in Tr$, $s'\in S$

但し, $s'=<q_1,q_2',c_{12},c_{21}>$ □

ところで, プロトコルの仕様によっては, これらのルールの適用が有限回で終了しない場合があり得る. 例えば, プロセスP_1の中に, ある状態から

L_1で出力を出して同じ状態に戻る遷移が定義されていると，rule Dが無限回適用できる．そこで，以下本論文ではこれらのルールの適用が有限回で終了する場合のみを対象とする．従って，以下ではSSGを有限グラフとして扱う．

SSG上において，初期状態からグラフ上のラベルを追うことによって，Pを構成する2つのプロセスを合成したシステム全体の各アクションの系列が得られる．このアクション系列にはシステム全体の各アクション間の前後関係が明確に表されている．これによって，同期の問題が解決される．

例えば，プロトコルPを構成するプロセスP_1, P_2より得られるアクションの系列がそれぞれ$a_1b_1c_1$および$d_2e_2f_2$であり，さらに上のSSGより得られるアクション系列が$a_1d_2e_2b_1f_2c_1$であるものとすると，アクションb_1はd_2, e_2の後に起こり，またf_2の前に起こることがわかる．

SSGより，3.4で与える方法によって，通信路が完全な場合(つまり送ったデータは必ず正しく届く場合)において完全性のある試験系列を求めることができる．

3.3 システム状態グラフの拡張

前節の手法によって，2つのプロセス間の通信路が完全な場合を想定したプロトコルでは，全てのアクション間の同期の問題が解決し，完全な試験系列を求めることができる．

ところが，2つのプロセス間の通信路が完全でない場合を想定したプロトコル，つまり，誤り制御機能をもったプロトコル等においては，その全てのアクションがSSG上に現れない．例えば，後述の図3-5のようなプロトコルでは，同図(a)における状態4から状態5への“T+timeout”でラベル付けされた遷移は，チャネル内でデータの消失があった場合を想定しており，前節の手法だけではこの遷移を含む正常なアクション系列のすべてがSSG上に現れない．ここで，正常なアクション系列とは，初期システム状態から始まり，2つのプロセスが定められた最終状態にありかつチャネルの状態が両方向のチャネルとも空であるようなシステム状態(これを最終システム状態と呼ぶ)に至る系列をいう．このため，このSSGより完全な試験系列を求めることができない．

そこで，ここでは通信路の誤りのうちデータ損失を対象とし，この誤りがある場合においても完全性のあるSSGを生成することができるようにSSG生成ルールを拡張する．この拡張によって得られるSSG，拡張SSG(ESG

: Extended System states Graph)より，次の3.4で与える方法によって，通信路にデータ損失の誤りがある場合においても完全性のある試験系列を求めることができる．

【 ESG生成規則 】

プロトコル$P=<<Q_1,A_1,Tr_1,o_1,F_1>,<Q_2,A_2,Tr_2,o_2,F_2>>$が与えられたとき，$A=A_1 \cup A_2 \cup \{\tau\}$と rule A ～ rule I 及び次の ruleJ , ruleK により得られる最小の集合S,Tr 及び初期システム状態s_0からなるSSG $G_E=<S,A,Tr,s_0>$をPに関するESGと定める．

<u>rule J</u> (チャネル12中のデータ消失)

$s=<q_1,q_2,c_{12},c_{21}> \in S,\ c_{12} \neq \emptyset$

$\vdash\ (s,\tau,s') \in Tr,\ s' \in S$

但し，$s'=<q_1,q_2,remain(c_{12}),c_{21}>$

<u>rule K</u> (チャネル21中のデータ消失)

$s=<q_1,q_2,c_{12},c_{21}> \in S,\ c_{21} \neq \emptyset$

$\vdash\ (s,\tau,s') \in Tr,\ s' \in S$

但し，$s'=<q_1,q_2,c_{12},remain(c_{21})>$ □

ここで，τはチャネル中のデータが消失したことを示す空アクションである．試験実施時には，テスタはこのアクションを起こすか他の入力を与えるかを決定できるものとする．

3.4 相互接続試験系列の導出

本節では，3.3で得られたESGより，相互接続試験系列を導出する方法について述べる．

第2章で述べたように，適合性試験に関する試験系列の生成法に関して，従来行われてきた研究として，TT法[Nait 81]，PW法[Chow 78]，DS法[Gone 70]，UI法[Sabn 88]等がある．これらは，いづれもMealy型FSMをモデルとし，入力に一定の制限をおいた上で，Mealy型FSMの重要な性質である出力関数および遷移関数の一方又は両方を調べることができる手法である[Sidh 89]．

本章で対象とする相互接続試験においても，2つの試験対象とその間のチャネルを合成したシステム全体の振る舞いを調べるためにこれらの手法を応用することによって，有用な試験系列を得ることができると考えられる．ところが，3.3で得られたESGは，Mealy型FSMとは異なるため，これに“直接”上記の手法を適用することはできない．そこで，ここではESGから，上記の手法の制限のうち共通的なものである「完全定義」「最小」「強連結」の3つの条件を満たしているようなMealy型FSMを導出し，このMealy型FSMに上記の手法の一つを適用して試験系列を導出する方法を提案する．

以下に，ESGからMealy型FSMを導出する方法を示し，Mealy型FSMの最小化の方法を適用し，さらに簡単な仮定をおくことによりこのMealy型FSM

が最小, 完全定義及び強連結の各条件を満たすことを述べる.

ESGとMealy型FSMの異なる主要な点は, 入出力の区別に関してである. すなわち, 前者が入力と出力を形の上で区別せず, どちらも一つのアクションとしてとらえているのに対して, 後者は入力と出力を区別し, ある状態からの遷移は入力によってはじめて起こり, その遷移の過程で出力を行うという形で表されることである. つまり, Mealy型FSMの上では現れない状態(例えば, 入力を行った後でかつ出力をする前の状態など)が, ESG上に存在することになる. そこで, ESGからMealy型FSMを導出するためには, 基本的には, 前者の状態のうち後者に存在するもののみを取り出して(このことを, ここでは状態の抽出と呼ぶ), グラフを再構成すればよい. ここで, 抽出された状態をexplicitな状態と呼び, 残りの状態をimplicitな状態と呼ぶ. また, 再構成されたグラフを状態遷移グラフ(STG : States Transition Graph)と呼ぶ. STGを次のように定める.

【 定義3-4 】STG

STG G_F は4項組 $< S_{exp} \cup S_{imp}, A, Tr', s_0 >$ である. ここで,

S_{exp} : explicitな状態の集合

S_{imp} : implicitな状態の集合

A : 入出力アクションの集合

Tr' : 遷移関係　$Tr' \subseteq (S_{exp} \cup S_{imp}) \times A \times (S_{exp} \cup S_{imp})$

s_0：初期状態 □

STG $G_F = < S_{exp} \cup S_{imp}, A, Tr', s_0 >$において，$(s_1,e,s_2) \in Tr'$であるとき，これを“$s_1 — e \rightarrow s_2$”と表す．

次に，ESGから状態を抽出し，STGを導出する方法を与える．

【 STG導出法 】

ESG $G_E = <S,A,Tr,s_0>$に対し，以下によって求められるSTG $G_F = < S_{exp} \cup S_{imp}, A, Tr', s_0 >$を$G_E$に関するSTGと定める．

$S_{exp} = \{s \in S | \exists s', a\ (s,a,s') \in Tr\} \cup \{s \in S | \exists s'\ (s,\tau,s') \in Tr\} \cup \{s_0\}$

ただし，$a = p + m$ or $Tk +$ timeout,

$p \in SAPk, m \in M, M = M_{kj} \cup M_{jk} \cup M_{ko} \cup M_{ki}$

$S_{imp} = \{ s_{imp} \mid s \in S - S_{exp} \}$

ただし，s_{imp}はsに対応するimplicitな状態を表す

$Tr' \subseteq (S_{exp} \cup S_{imp}) \times A \times (S_{exp} \cup S_{imp})$

$Tr' = \{ (s,e,s') \mid (s,e,s') \in Tr, s,s' \in S_{exp} \}$

$\cup \{ (s,e,s_{imp}') \mid (s,e,s') \in Tr, s \in S_{exp}, s' \in S - S_{exp} \}$

$\cup \{ (s_{imp},e,s') \mid (s,e,s') \in Tr, s \in S - S_{exp}, s' \in S_{exp} \}$

$\cup \{ (s_{imp},e,s_{imp}') \mid (s,e,s') \in Tr, s,s' \in S - S_{exp} \}$ □

ここで，G_Eの初期システム状態s_0は，常にexplicitな状態であるものとする．また，τはテスタが制御できるという仮定のもとに，これを1つの入力と

して扱う.

ところで, 一般にこのSTGにはimplicitな状態における合流及び分岐が存在する. 一方, Mealy型FSMは入力の前を状態と見なすので, implicitな状態における合流及び分岐が存在しないSTG(このようなSTGを標準形STG(NSG : Normal States transition Graph)と呼ぶ)であればこれをMealy型FSMに変換しやすい. そこで, 一般のSTGを標準形に変形する. 変形は, 次に示す方法により, implicitな状態における合流及び分岐を「開く」ことによって行われる. Mealy型FSMでは入力の前のみを状態と見なし, それ以外の状態を無視するので, STGをMealy型FSMに変換した場合, STG上のimplicitな状態すなわち入力の前以外の状態における合流及び分岐を開くことによってその意味はかわらない.

以下に, 一般のSTGを標準形に変形する方法を示す.

【 STG標準化手順 】

標準化は, STG上に存在するimplicitな状態における合流及び分岐に対して, それぞれ図3-1及び図3-2に示す方式に従って行う. すなわち, 図3-1(a)のようなimplicitな状態における合流が存在した場合, 新たな(implicitな)状態を設け, 合流状態を複数に分ける(同図(b)). このような変形をimplicitな状態における合流が存在しなくなるまで繰り返す. 明らかに, 上記の一回の変形においてimplicitな状態の数は同じかまたは一つ減少し, またSTG上の

implicitな状態の数は有限であるので，この変形の繰り返しは有限回で終了する．同様に，図3-2(a)のようなimplicitな状態における分岐が存在した場合，新たな(implicitな)状態を設け，分岐状態を複数に分ける(同図(b))．このような変形をimplicitな状態における分岐が存在しなくなるまで繰り返す．上と同様に，この場合も変形の繰り返しは有限回で終了する． □

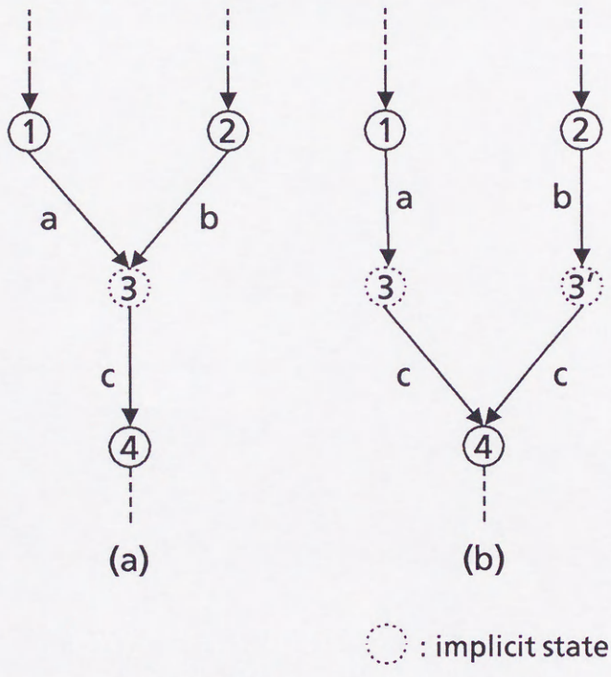


図3-1　implicitな状態における合流の変形

以上の様にして得られるNSGにおいて，次の記法を定める．すなわち，NSG $G_N = < S_{exp} \cup S_{imp}', A, Tr'', s_0 >$ において2つのexplicitな状態s_0, $s_n \in S_{exp}$を結ぶ遷移の系列

$s_0 — a_1 \rightarrow s_1$, $s_1 — a_2 \rightarrow s_2$, …, $s_{n-1} — a_n \rightarrow s_n$

(ただし，s_1, …, $s_{n-1} \in S_{imp}$, $n \geqq 1$)

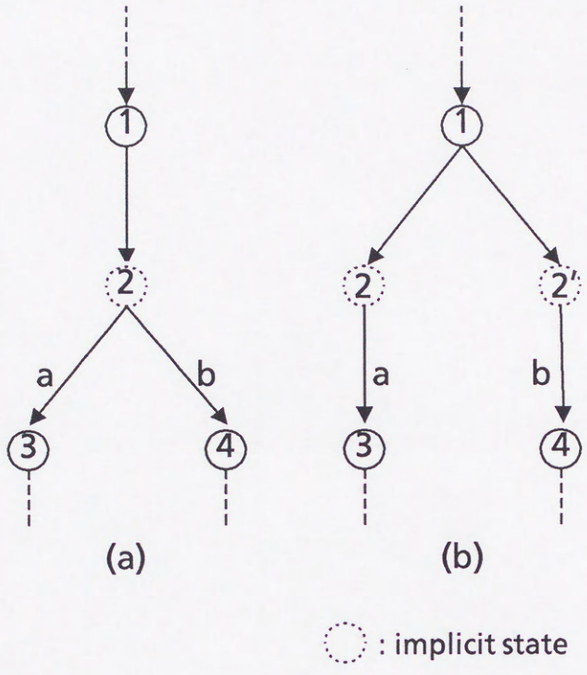


図3-2　implicitな状態における分岐の変形

が存在するとき，これを

$s_1 — a_1 a_2 \cdots a_n \rightarrow s_2$

と表す．

NSGは，STGの作り方より，各explicitな状態から入力のラベルを持った遷移が存在することが，また上記の変形により，分岐はexplicitな状態のみで起こることが，それぞれ保証されている．そこで，NSG $G_N = < S_{exp} \cup S_{imp}', A, Tr'', s_0 >$ 上の各explicitな状態をMealy型FSMの状態に対応させることにより，次のようにしてMealy型FSMを導出する．

【 Mealy型FSM導出法 】

NSG $G_N = < S_{exp} \cup S_{imp}', A, Tr'', s_0 >$ に対して，以下によって求められる Mealy型FSM $M_M = <Q, I, O, \omega, \delta>$ を G_N に関する Mealy型FSMと定める．

$Q = S_{exp}$

$I = \{e | e \in A, e$は入力アクション$\} \cup \{\tau\}$

$O = \{e | e \in A, e$は出力アクション$\}$

$\omega(s_1, a_1) = a_2 a_3 \cdots a_n \quad (\exists s_2 \in Q\ s_1 — a_1 a_2 \cdots a_n \rightarrow s_2,\ a_1 \in I, a_2 \cdots a_n \in O, n \geqq 2)$

$\omega(s_1, a_1) = \varepsilon \quad (\exists s_2 \in Q\ s_1 — a_1 \rightarrow s_2,\ a_1 \in I)$

$\omega(s_1, \varepsilon) = a_1 a_2 \cdots a_n \quad (\exists s_2 \in Q\ s_1 — a_1 a_2 \cdots a_n \rightarrow s_2,\ a_1 \cdots a_n \in O,\ n \geqq 1)$

$\delta(s_1, a_1) = s_2 \quad (\exists s_2 \in Q\ s_1 — a_1 a_2 \cdots a_n \rightarrow s_2,\ a_1 \in I,\ a_2 \cdots a_n \in O,\ n \geqq 1)$

$\delta(s_1, \varepsilon) = s_2 \quad (\exists s_2 \in Q\ s_1 — a_1 a_2 \cdots a_n \rightarrow s_2,\ a_1 \cdots a_n \in O,\ n \geqq 1)$ □

すなわち，NSG $G_N = < S_{exp} \cup S_{imp}', A, Tr'', s_0 >$ 上の各explicitな状態をMealy型FSMの状態に，入力アクションの集合をIに，出力アクションの集合をOにそれぞれ対応させる．さらに，隣り合う2つのexplicitな状態を結ぶアクションの系列より，Mealy型FSMの遷移が得られる(図3-3)．ただし，図3-4(a)の"*1—in1→2*"のように，隣り合う2つのexplicitな状態を結ぶアクションの系列中に出力アクションがない場合，空出力εを補う(同図(b))．又，同図(a)の"*1—out2out3→4*"のように，アクションの系列中に入力アクションがない場合，遷移の先頭に空の入力εを加える(同図(b))．ここで，εはテスタより制御可能であるものとする．

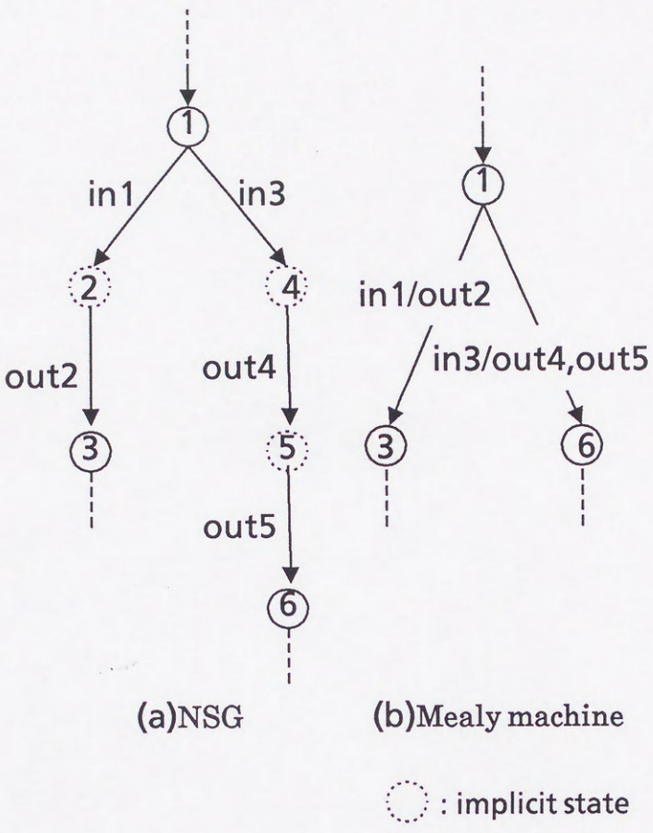


図3-3　NSGからのMealy型FSMの導出

以上のようにして得られたMealy型FSMは，FSMの最小化アルゴリズムを適用することによって，最小のものが得られる[Toma 83]．また，後に図3-7に示すように，定義されない入力に対しては，空出力を出して同じ状態に戻る遷移を仮定することによって，このMealy型FSMは完全定義であると見なすことができる．また，各状態について，初期状態に戻るためのリセット系列を仮定することにより，強連結の条件を満たすと考えることができる．

これらのことから，上で導出されたMealy型FSMが，従来の試験系列の生成法であるTT法，PW法，DS法，UI法に共通する制限条件を満たすことが示

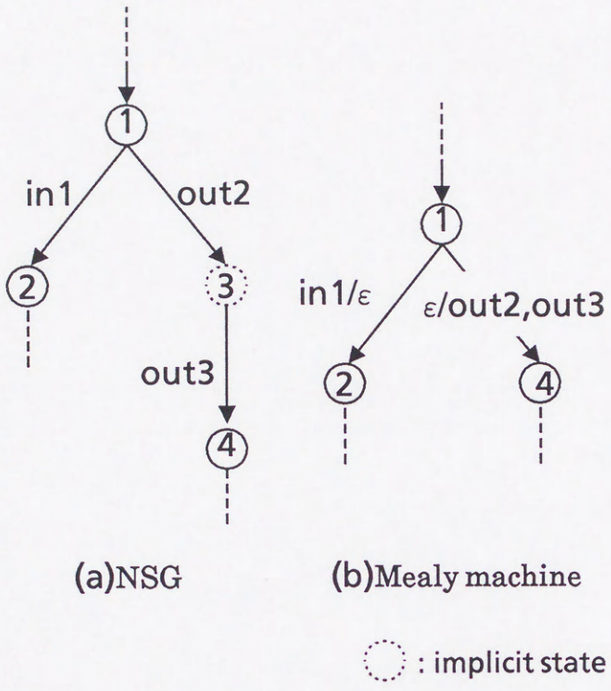


図3-4　出力/入力遷移がない場合の
Mealy型FSMの導出

された(ただし, DS法に関してはさらに判定系列が存在することが必要).

そこで, このMealy型FSMについて, これらの手法の一つを適用して試験系列を得ることができる. 本論文では, 3.5でUI法を適用した例を示す. UI法は, UIO(Unique Input and Output)シーケンスと呼ばれる各状態をユニークに識別する系列をもとに, 各遷移についてその出力と遷移先状態を調べる手法である.

3.5 適用例

以上のことを実際の仕様例に適用した例を示す. 図3-5のような仕様を考える.

この仕様例において, 例えば, プロセスP_2(同図(b))の"L2+mess"でラベル付けされた状態1から状態2への遷移に注目する. これに対して, 3.3で示したESG生成規則のrule Gを適用することにより, システム状態<4,1,mess,E>から<4,2,E,E>への"L2+mess"でラベル付けされた遷移が推論される. このようにして図3-5の仕様から, ESG生成規則を適用することにより図3-6のESGが得られる. 同図において, 二重線で示される遷移は, 3.3で拡張したルールによって推論される遷移である.

次に, 図3-6のESGにおいて, 下線を引いた状態がexplicitな状態であり, 他の状態がimplicitな状態であると考えることにより, これは3.4で示したSTG導出法により状態の抽出をしたSTGであると見なすことができる. ここで, 例えば同図の状態<4,1,mess,E>は, 入力ラベル"L2+mess"を持った遷移が存在するため, explicitな状態となる. 一方, 状態<4,2,E,E>は, 出力ラベル"U2−mess"を持った遷移のみ存在し, 入力ラベルを持った遷移が存在しないため, implicitな状態となる.

ここで, 図3-6のimplicitな状態<5,1,E,E>を見ると, 状態<4,1,E,E>か

らの遷移と状態<4,1,E,nak>からの遷移の，2つの遷移の合流が存在する．そこで，図3-1の方法によりこれを変形する．このようにして3.4で示したSTG標準化手順により図3-6のSTGを変形し，図3-7のNSGを得ることができる．ただし，同図においてimplicitな状態については省略している．

さらに，図3-7中の

<4,1,mess,E>—L2＋mess→<4,2,E,E>，

<4,2,E,E>—U2－mess→<4,3,E,E>

(ただし，図中で状態<4,2,E,E>は省略)

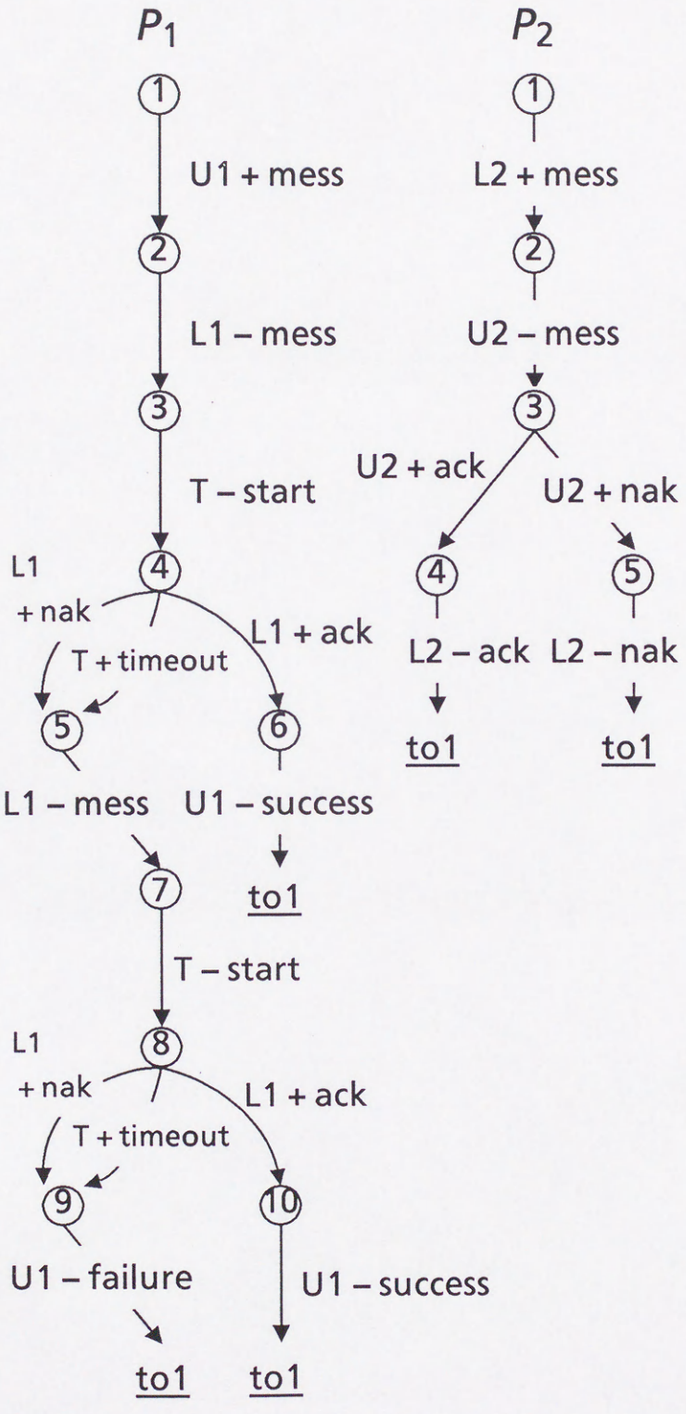
の遷移の系列から，3.4で与えたMealy型FSM導出法により

<4,1,mess,E> —L2＋mess / U2－mess→ <4,3,E,E>

という遷移が導かれる．ここで，“s_1—e_i / e_o→s_2”は，入力e_iによって状態s_1から出力e_oを出して状態s_2へ移る遷移を表す．このようにして図3-7のNSGから導出されるMealy型FSMを最小化したものは図3-8のようになる．ここで，図中の“-”は，空出力を出して同じ状態に戻る遷移を表す．

最後に，このMealy型FSMについて，従来の試験系列の生成法の一つを適用する．ここでは，UI法を適用した例を示す．UIOシーケンスは図3-9のようになる．ここで，例えば同図の1行目より，入力“U1＋mess”を与えたときの出力が“L1－mess”であれば，この入力を与える前の状態は“1(<1,1,E,E>)”であることがわかる．このUIOシーケンスをもとに，図3-10のような試験系列

が得られる．ここで，rはリセット系列を表す．例えば同図(a)の5，6行目は，図3-8(a)の状態2(<4,3,E,E>)から入力“U2+nak”による状態4(<4,1,E,nak>)への遷移を調べる系列を表す．ここで，“U1+mess, ε, L2+mess”の部分は初期状態から状態2への遷移を，“U2+nak”の部分は目的の遷移をそれぞれ表す．また，“L1+nak”の部分は目的の遷移の遷移先状態4を調べるUIOシーケンスである．



(a) (b)

図3-5　プロトコル例

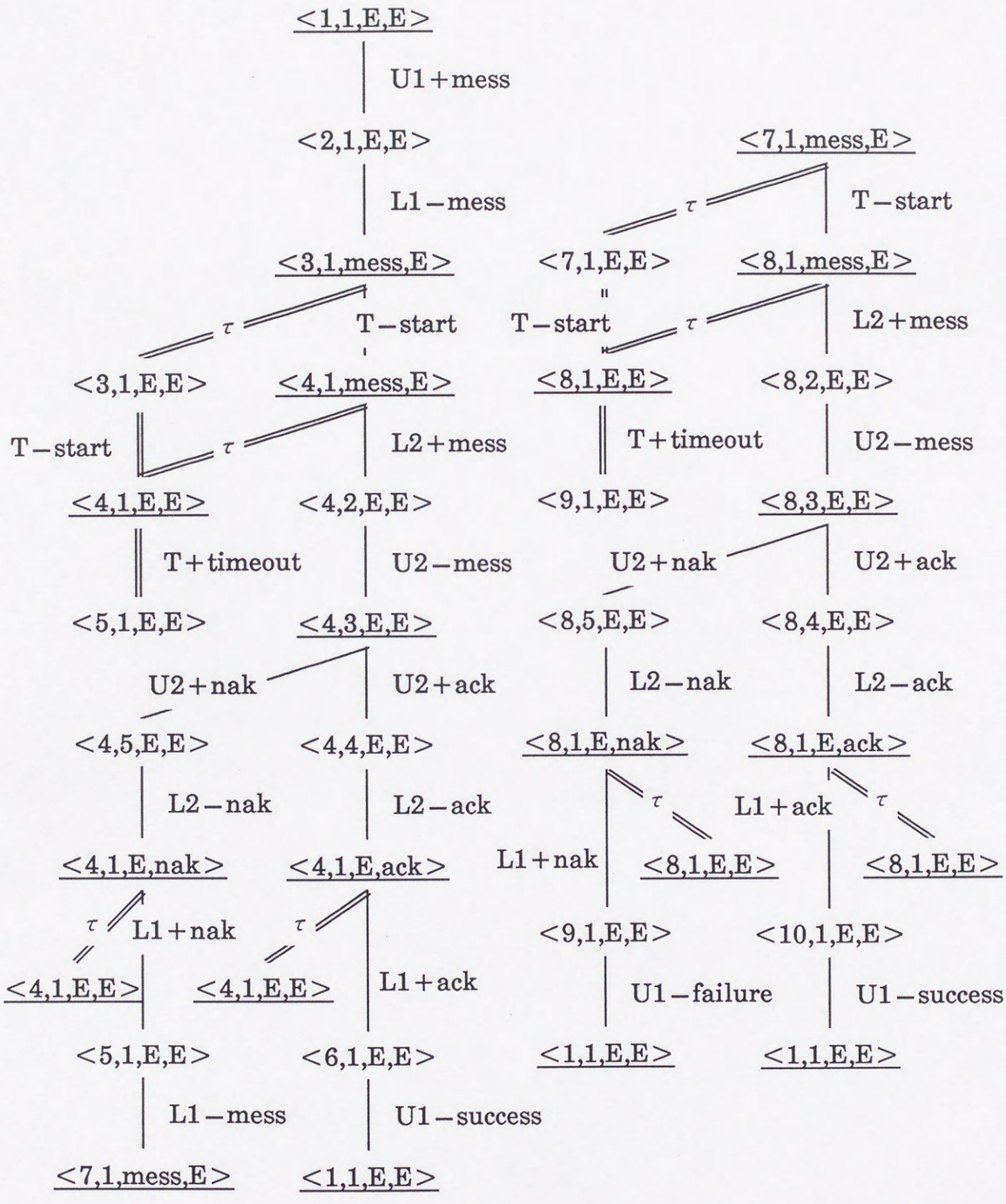

<1,1,E,E>
U1+mess
<2,1,E,E>
L1−mess
<3,1,mess,E>
τ
T−start
<3,1,E,E>
<4,1,mess,E>
T−start
τ
L2+mess
<4,1,E,E>
<4,2,E,E>
T+timeout
U2−mess
<5,1,E,E>
<4,3,E,E>
U2+nak
U2+ack
<4,5,E,E>
<4,4,E,E>
L2−nak
L2−ack
<4,1,E,nak>
<4,1,E,ack>
τ
L1+nak
τ
<4,1,E,E>
<4,1,E,E>
L1+ack
<5,1,E,E>
<6,1,E,E>
L1−mess
U1−success
<7,1,mess,E>
<1,1,E,E>
<7,1,mess,E>
τ
T−start
<7,1,E,E>
<8,1,mess,E>
T−start
τ
L2+mess
<8,1,E,E>
<8,2,E,E>
T+timeout
U2−mess
<9,1,E,E>
<8,3,E,E>
U2+nak
U2+ack
<8,5,E,E>
<8,4,E,E>
L2−nak
L2−ack
<8,1,E,nak>
<8,1,E,ack>
τ
L1+ack
τ
L1+nak
<8,1,E,E>
<8,1,E,E>
<9,1,E,E>
<10,1,E,E>
U1−failure
U1−success
<1,1,E,E>
<1,1,E,E>

図3-6　ESG, STG

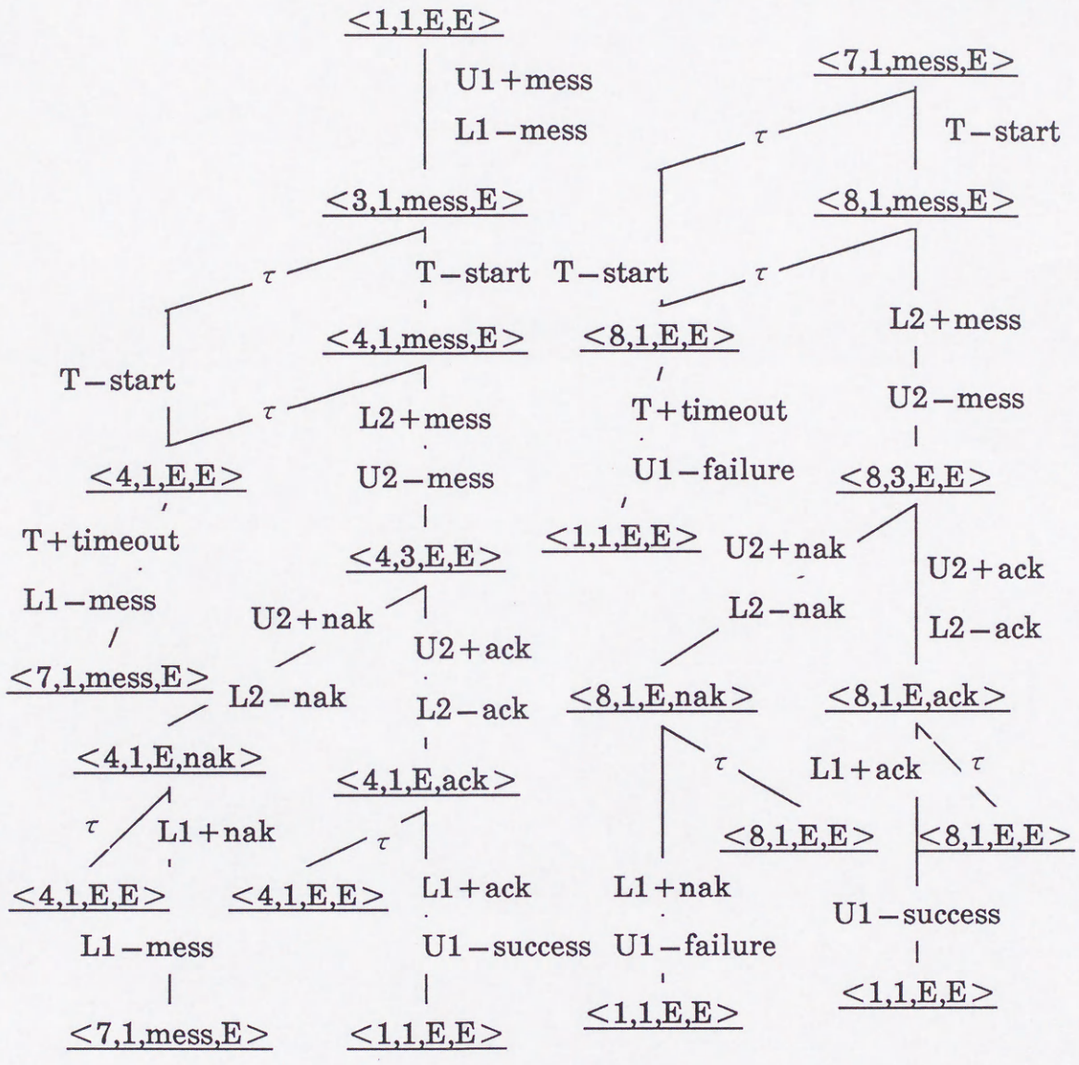

<1,1,E,E>
U1+mess
L1−mess
<3,1,mess,E>
τ
T−start
T−start
<4,1,mess,E>
τ
L2+mess
T−start
<4,1,E,E>
U2−mess
T+timeout
<4,3,E,E>
L1−mess
U2+nak
U2+ack
<7,1,mess,E>
L2−nak
L2−ack
<4,1,E,nak>
<4,1,E,ack>
τ
L1+nak
τ
<4,1,E,E>
<4,1,E,E>
L1−mess
L1+ack
U1−success
<7,1,mess,E>
<1,1,E,E>
<7,1,mess,E>
τ
T−start
T−start
<8,1,mess,E>
τ
<8,1,E,E>
L2+mess
T+timeout
U2−mess
U1−failure
<8,3,E,E>
<1,1,E,E>
U2+nak
U2+ack
L2−nak
L2−ack
<8,1,E,nak>
<8,1,E,ack>
τ
L1+ack
τ
<8,1,E,E>
<8,1,E,E>
L1+nak
U1−success
U1−failure
<1,1,E,E>
<1,1,E,E>

図3-7　NSG

states	U1＋mess	L2＋mess	U2＋ack	U2＋nak	L1＋ack
1	L1－mess / 12	-	-	-	-
2	-	-	L2－ack / 5	L2－nak / 4	-
3	-	-	L2－ack / 7	L2－nak / 6	-
4	-	-	-	-	-
5	-	-	-	-	U1－success /1
6	-	-	-	-	-
7	-	-	-	-	U1－success /1
8	-	U2－mess / 2	-	-	-
9	-	U2－mess / 3	-	-	-
10	-	-	-	-	-
11	-	-	-	-	-
12	-	-	-	-	-
13	-	-	-	-	-

1 : <1,1,E,E>　　2 : <4,3,E,E>　　3 : <8,3,E,E>
4 : <4,1,E,nak>　　5 : <4,1,E,ack>　　6 : <8,1,E,nak>
7 : <8,1,E,ack>　　8 : <4,1,mess,E>　　9 : <8,1,mess,E>
10 : <4,1,E,E>　　11 : <8,1,E,E>　　12 : <3,1,mess,E>
13 : <7,1,mess,E>

図3-8(a)　Mealy型FSMの最小形(その1)

states	L1＋nak	T＋timeout	τ	ε
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	L1－mess / 13	-	ε / 10	-
5	-	-	ε / 10	-
6	U1－failure /1	-	ε / 11	-
7	-	-	ε / 11	-
8	-	-	ε / 10	-
9	-	-	ε / 11	-
10	-	L1－mess / 13	-	-
11	-	U1－failure / 1	-	-
12	-	-	T－start / 10	T－start / 8
13	-	-	T－start / 11	T－start / 9

1 : <1,1,E,E>　　2 : <4,3,E,E>　　3 : <8,3,E,E>
4 : <4,1,E,nak>　　5 : <4,1,E,ack>　　6 : <8,1,E,nak>
7 : <8,1,E,ack>　　8 : <4,1,mess,E>　　9 : <8,1,mess,E>
10 : <4,1,E,E>　　11 : <8,1,E,E>　　12 : <3,1,mess,E>
13 : <7,1,mess,E>

図3-8(b)　Mealy型FSMの最小形(その2)

state	UIO
1	U1＋mess / L1－mess
2	U2＋nak / L2－nak L1＋nak / L1－mess
3	U2＋nak / L2－nak L1＋nak / U1－failure
4	L1＋nak / L1－mess
5	L1＋nak / ε L2＋mess / ε τ / ε T＋timeout / L1－mess
6	L1＋nak / U1－failure
7	L1＋nak / ε L2＋mess / ε τ / ε T＋timeout / U1－failure
8	L2＋mess / U2－mess U2＋nak / L2－nak L1＋nak / L1－mess
9	L2＋mess / U2－mess U2＋nak / L2－nak L1＋nak / U1－failure
10	T＋timeout / L1－mess
11	T＋timeout / U1－failure
12	τ / T－start T＋timeout / L1－mess
13	τ / T－start T＋timeout / U1－failure

図3-9　UIOシーケンス

r,

U1 + mess,τ,T + timeout

r,U1 + mess,ε,L2 + mess,

U2 + ack,L1 + nak,L2 + mess,τ,T + timeout

r,U1 + mess,ε,L2 + mess,

U2 + nak,L1 + nak

r,U1 + mess,τ,T + timeout,ε,L2 + mess,

U2 + ack,L1 + nak,L2 + mess,τ,T + timeout

r,U1 + mess,τ,T + timeout,ε,L2 + mess,

U2 + nak,L1 + nak

r,U1 + mess,ε,L2 + mess,U2 + nak,

L1 + nak,τ,T + timeout

r,U1 + mess,ε,L2 + mess,U2 + nak,

τ,T + timeout

r,U1 + mess,ε,L2 + mess,U2 + ack,

L1 + ack,U1 + mess

r,U1 + mess,ε,L2 + mess,U2 + ack,

τ,T + timeout

r,U1 + mess,τ,T + timeout,ε,L2 + mess,U2 + nak,

L1 + nak,U1 + mess

r,U1 + mess,τ,T + timeout,ε,L2 + mess,U2 + nak,

τ,T + timeout

r,U1 + mess,τ,T + timeout,ε,L2 + mess,U2 + ack,

L1 + ack,U1 + mess

図3-10(a)　試験系列(その1)

r,U1 + mess,τ,T + timeout,ε,L2 + mess,U2 + ack,

τ,T + timeout

r,U1 + mess,ε,

L2 + mess,U2 + nak,L1 + nak

r,U1 + mess,ε,

τ,T + timeout

r,U1 + mess,τ,T + timeout,ε,

L2 + mess,U2 + nak,L1 + nak

r,U1 + mess,τ,T + timeout,ε,

τ,T + timeout

r,U1 + mess,τ,

T + timeout,τ,T + timeout

r,U1 + mess,τ,T + timeout,τ,

T + timeout,U1 + mess

r,U1 + mess,

τ,T + timeout

r,U1 + mess,

ε,L2 + mess,U2 + nak,L1 + nak

r,U1 + mess,τ,T + timeout,

τ,T + timeout

r,U1 + mess,τ,T + timeout,

ε,L2 + mess,U2 + nak,L1 + nak

図3-10(b)　試験系列(その2)

3.6 まとめ

本章では，相互接続試験における試験系列の生成の手法を提案した．ここでは，2つの試験対象とその間のチャネルを合成したシステム全体の振る舞いを調べるために，システム状態グラフを導入した．そして，2つのFSMとして記述されたプロトコル仕様よりシステム状態グラフを生成するルールを与えた．このルールにより得られたシステム状態グラフは，相互接続試験時に動作すべきシステム全体の振る舞いを表しており，ここには2つのエンティティのアクション間の相互の時間的前後関係が明確に表されている．このことにより，チャネルが完全であることを想定したプロトコル仕様の場合において，相互接続試験における同期の問題が解決することを示した．次に，チャネルにエラーがある場合を想定したプロトコル仕様に対しても適用できるようシステム状態グラフ生成ルールを拡張した．拡張されたルールにより得られた拡張システム状態グラフは，チャネルエラーを想定したプロトコル仕様の場合でも，動作すべきシステム全体の振る舞いを表し，これによって，この場合でも同期の問題が解決することを示した．さらに，拡張システム状態グラフを変形することによって，従来の適合性試験に関する試験系列生成法が適用可能になるようにし，これらの手法を用いて試験系列を生成できることを示した．これによって，従来の技術を有効に利用し，相互接

続試験における有効な試験系列の生成が行えると考えられる．本手法を用いることによって，相互接続試験における試験系列生成のコストの削減，試験の信頼性の向上などが期待される．

今後の課題としては，長さ，計算量などの効率に関することを含め，具体的に本手法が有効に適用可能なプロトコルの種別，規模，複雑さなどに関する研究の他，支援システムの構築等がある．

第4章　LOTOS仕様からの試験系列の生成法

4.1 はじめに

本章では，イベントの時間順序に基づくFDTとしてLOTOSを対象とした形式的な適合性試験系列の生成手法を与える．

まず4.2では，本手法の対象となるLOTOSについて紹介する．特にLOTOSの形式的意味を与えるLTSの等価性について述べる．4.3では変数付きLTSであるVTSを導入し，その導出体系を与える．また，VTSの意味を与える具現化規則を定め，ここで与える導出体系がLOTOSで定められているLTSの導出体系と矛盾がないことを示す．4.4では4.3で導入したVTSを用いたLOTOS仕様からの試験系列の生成法を提案し，その手法を論じる．4.5では，4.4で提案した手法の適用例を示す．4.6は本章のまとめである．

4.2 LOTOSとその形式的意味

本節では，LOTOSの概要を紹介し，LOTOSの形式的意味を与えるLTSとその等価性について述べる．LOTOSの詳細については[ISO 89a]を参照されたい．

4.2.1 LOTOSの概要[ISO 89a] [Taka 90]

LOTOSは基本的に次のような考え方に基づいている．

「システムは，そのシステムとやりとりする外部から観測可能なイベントの時間順序を規定することによって仕様化可能である」

LOTOSでは，システムの記述を以下で述べる“プロセス”の観点から行い，外部から観測される振る舞いとして通信能力を記述することでシステムの仕様記述を行う．プロセスの中で出現するデータの定義には，データ型の等式を用いた記述(抽象データ型の代数仕様記述)が使用される．上記のプロセスの振る舞い(動作)の記述はCCS[Miln 80]に基礎をおいており，記述された仕様の解析に有効な理論的枠組みを提供する．データの部分は抽象データ型[Inag 84]の記述言語ACTONEに基づいている．

LOTOSにおいては，記述の対象となる単位をプロセスと呼び，システムはプロセスあるいは通信するいくつかのサブプロセスから成るプロセスと見なされる．他のプロセスとの通信能力を記述することがプロセスの定義に

なり，システム全体の仕様はプロセス定義の階層構造によって表される．プロセスとは，その環境における他のプロセスと通信を行う抽象的な実体を指す．プロセス間の通信は，“ゲート”と呼ばれるインタラクション-ポイントで起こり，インタラクションの基本単位は，“イベント”あるいは“アクション”と呼ばれる．イベントは，“アトミック”で“同期的”なインタラクションである．イベントがアトミックであるというのは，

- 瞬時発生的であり，
- 構造化できず，
- 他のイベントと時間的なオーバーラップがない，

ということを意味する．また，イベントが同期的であるというのは，二つ以上の通信するプロセスが同時にその発生に巻き込まれるということを意味する．イベントにおいては，必要があればデータ(の値)の交換が行われ，このデータは抽象データタイプ(ADT : Abstract Data Type)[Gutt 77] [Gutt 78] として代数的に記述される[Gogu 78] [Inag 84] ．ここで，“抽象”的とはデータの表現の仕方と独立であり，インプリメンテーションに関する制約を受けないことを意味する．データの交換を伴わない通信においては，ゲートとイベントは一致し，単に同期を表す．

プロセスの静的側面は，図4-1に示すように，ブラックボックスと見なされ，内部のメカニズムは観測不能であり，内部で発生するイベントはすべて

内部イベント“i”によって表される．LOTOSの特徴は，外部的に観測可能なイベントや内部イベント“i”の時間的順序を陽に記述することである．

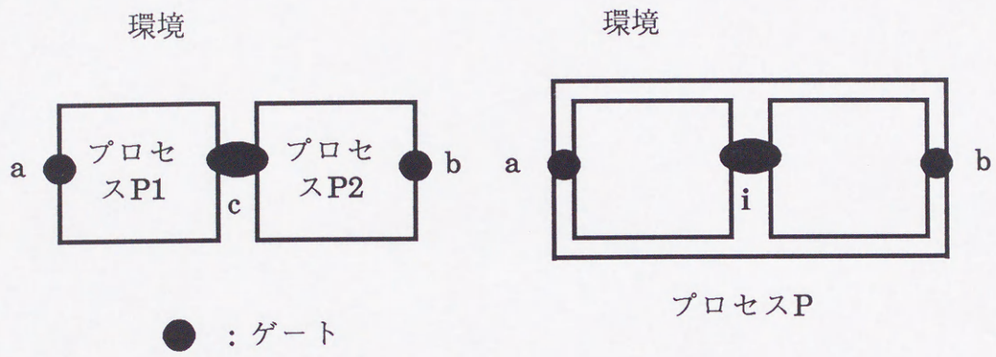


図4-1　LOTOSにおけるプロセスの概念

LOTOSにおいて，プロセスの定義は次のように表される．

process ＜process-identifier＞ ＜parameter-part＞ :=

＜behaviour-expression＞

endproc

プロセスの振る舞いは，各ゲートで起こる観測可能なイベントの系列を動作式(behaviour expression)として記述することによって表現される．

動作式の構成規則はCCS [Miln 80]の理論に基づいており，動作式を記述するためのオペレーションには次のものがある．詳細については文献を参照されたい．

(a) action prefix (; で表す)

: a;Bは, イベント a が起こり, その後の動作がBで表されることを意味する. ここで, Bは動作式である.

(b) choice ([] で表す)

: B_1 [] B_2 は, B_1 と B_2 のどちらかが選択されることを意味する. ここで, B_1 と B_2 は動作式である.

(c) inaction (**stop** で表す)

: プロセスが停止することを表すオペレーションである.

(d) parallel composition ($|[g_1, \cdots, g_n]|$ で表す)

: $B_1 |[g_1, \cdots, g_n]| B_2$ は, B_1 と B_2 がゲート $g_1, \cdots, g_n$ で同期をとりながら, 並列に動作することを意味する. ここで, B_1 と B_2 は動作式である.

(e) 他に, hiding, instantiation, guarding, enabling, disabling, 等のオペレーションがある.

動作式によって表されたプロセスの振る舞いの様子は, 図4-2のような木構造によって表すことができる.

4.2.2 ラベル付き遷移システム(LTS)[ISO 89a]

LOTOS仕様の形式的モデルは, プロセスの動的振る舞いの記述である動作式のモデルのLTSである. 動作式が与えられると, LOTOSのオペレーショナル-セマンティックス(操作的意味)を与える公理と推論規則から定義さ

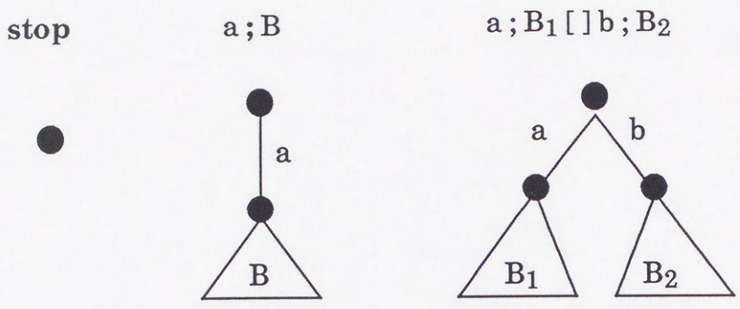


図4-2　アクション木

れる遷移関係に基づいて，LTSを生成することができる．表4-1に，Basic-LOTOSに関する公理と推論規則を紹介する．ここで，Basic-LOTOSとは，データタイプを省略したLOTOSのサブセットで，プロセス間の同期のみを扱う．これに対して，データタイプを含む一般のLOTOSをFull-LOTOSということがある．Full-LOTOSに関するLTSの生成方法については文献[ISO 89a]を参照されたい．

(action-prefix) $a\,;B \xrightarrow{-a\rightarrow} B$

(choice)

$$\frac{B1 \xrightarrow{-a\rightarrow} B1'}{B1\ [\,]\ B2 \xrightarrow{-a\rightarrow} B1'} \qquad \frac{B2 \xrightarrow{-a\rightarrow} B2'}{B1\ [\,]\ B2 \xrightarrow{-a\rightarrow} B2'}$$

(process instantiation)

$$\frac{Bp[g_1/g_1', \cdots, g_n/g_n'] \xrightarrow{-a\rightarrow} B'}{p[g_1, \cdots, g_n] \xrightarrow{-a\rightarrow} B'}$$

但し, "**process** $p[g_1', \cdots, g_n'] :=$ Bp **endproc**" をプロセス p の定義とする

(relabelling)

$$\frac{B \xrightarrow{-a\rightarrow} B'}{B[g_1/g_1', \cdots, g_n/g_n'] \xrightarrow{-a'\rightarrow} B'[g_1/g_1', \cdots, g_n/g_n']}$$

但し, $a \notin \{g_1', \cdots, g_n'\}$ の時 $a'=a$.
$a=g_i'$ $(1 \leq i \leq n)$ の時 $a'=g_i$.

(parallel composition)

$$\frac{B1 \xrightarrow{-a\rightarrow} B1',\ a \notin \{g_1, \cdots, g_n, \delta\}}{B1\ |[g_1, \cdots, g_n]|\ B2 \xrightarrow{-a\rightarrow} B1'\ |[g_1, \cdots, g_n]|\ B2} \qquad \frac{B2 \xrightarrow{-a\rightarrow} B2',\ a \notin \{g_1, \cdots, g_n, \delta\}}{B1\ |[g_1, \cdots, g_n]|\ B2 \xrightarrow{-a\rightarrow} B1\ |[g_1, \cdots, g_n]|\ B2'}$$

$$\frac{B1 \xrightarrow{-a\rightarrow} B1',\ B2 \xrightarrow{-a\rightarrow} B2',\ a \in \{g_1, \cdots, g_n, \delta\}}{B1\ |[g_1, \cdots, g_n]|\ B2 \xrightarrow{-a\rightarrow} B1'\ |[g_1, \cdots, g_n]|\ B2'}$$

$$\frac{B1\ |[\,]|\ B2 \xrightarrow{-a\rightarrow} B'}{B1\ |||\ B2 \xrightarrow{-a\rightarrow} B'} \qquad \frac{B1\ |[G]|\ B2 \xrightarrow{-a\rightarrow} B'}{B1\ ||\ B2 \xrightarrow{-a\rightarrow} B'}$$

但し, G はすべてのイベント (ゲート) のリスト

(hiding)

$$\frac{B \xrightarrow{-a\rightarrow} B',\ a \notin \{g_1, \cdots, g_n\}}{\textbf{hide}\ g_1, \cdots, g_n\ \textbf{in}\ B \xrightarrow{-a\rightarrow} \textbf{hide}\ g_1, \cdots, g_n\ \textbf{in}\ B'}$$

$$\frac{B \xrightarrow{-a\rightarrow} B',\ a \in \{g_1, \cdots, g_n\}}{\textbf{hide}\ g_1, \cdots, g_n\ \textbf{in}\ B \xrightarrow{-i\rightarrow} \textbf{hide}\ g_1, \cdots, g_n\ \textbf{in}\ B'}$$

(successful termination) **exit** $\xrightarrow{-\delta\rightarrow}$ **stop**

(enabling)

$$\frac{B1 \xrightarrow{-a\rightarrow} B1',\ a \neq \delta}{B1\ >>\ B2 \xrightarrow{-a\rightarrow} B1'\ >>\ B2} \qquad \frac{B1 \xrightarrow{-\delta\rightarrow} B1'}{B1\ >>\ B2 \xrightarrow{-i\rightarrow} B2}$$

(disabling)

$$\frac{B1 \xrightarrow{-a\rightarrow} B1',\ a \neq \delta}{B1\ [>\ B2 \xrightarrow{-a\rightarrow} B1'\ [>\ B2} \qquad \frac{B1 \xrightarrow{-\delta\rightarrow} B1'}{B1\ [>\ B2 \xrightarrow{-\delta\rightarrow} B1'}$$

$$\frac{B2 \xrightarrow{-a\rightarrow} B2'}{B1\ [>\ B2 \xrightarrow{-a\rightarrow} B2'}$$

表4-1　基本LOTOSの意味

4.2.3 等価性[ISO 89a] [Taka 90]

LOTOSにおける等価性は弱bisimulation等価と呼ばれ，これは“外部的に観測可能な有限回の動作(アクション)によって区別できないプロセスを同一視する”ことに基づいている．外部的に観測可能な動作はアクション系列の遷移関係“$=t\Rightarrow$”によって定式化される．

【 定義4-1 】アクション系列の遷移関係

$Sys = < S, A, T, s_0 >$ をLTSとする．

(1)遷移関係“$-t\rightarrow$”

$s,s' \in S$, $a_1,\cdots,a_n \in A$とし，tをアクションの系列$a_1 \cdots a_n$としたとき，関係$-t\rightarrow$を次のように定義する．

$s = s_0 - a_1 \rightarrow s_1, \cdots, s_{n-1} - a_n \rightarrow s_n = s'$であるような状態$s_0,\cdots,s_n \in S$が存在するとき，これを$s - t \rightarrow s'$で表す．特に，$n=0$に対しては$s - \varepsilon \rightarrow s$である．ここで，$\varepsilon$は空系列を表す．

(2)遷移関係“$=t\Rightarrow$”

$s,s' \in S$, $a_1,\cdots,a_n \in A-\{\mathrm{i}\}$, tを観測可能なアクションの系列$a_1 \cdots a_n \in (A-\{\mathrm{i}\})^*$とし，$\mathrm{i}^k$を$k(k \geqq 0)$個の内部アクションの系列としたとき，関係$=t\Rightarrow$を次のように定義する．

$s - \mathrm{i}^{k_0} a_1 \mathrm{i}^{k_1} \cdots a_n \mathrm{i}^{k_n} \rightarrow s'$であるようなアクションの系列$\mathrm{i}^{k_0} a_1 \mathrm{i}^{k_1} \cdots a_n \mathrm{i}^{k_n}$が存

在するとき，これを$s=t\Rightarrow s'$と表す．特に，$s-\mathbf{i}^k\rightarrow s'$を$s=\varepsilon\Rightarrow s'$で表し，またすべての$s$に対して$s=\varepsilon\Rightarrow s$とする． □

関係“$=t\Rightarrow$”を用いて，弱bisimulation関係を以下のように定義する．

【 定義4-2 】弱bisimulation関係

$Sys1=<S_1,A,T_1,s_{01}>$，$Sys2=<S_2,A,T_2,s_{02}>$ を任意のLTSとし，$S=S_1\cup S_2$とする．次のような状態の集合S上の二項関係$R\subseteq S\times S$を弱bisimulation関係という．

状態の組(s_1,s_2)が$(s_1,s_2)\in R$であるのは，任意の観測可能なアクションの系列$t\in(A-\{\mathbf{i}\})^*$に対して，次の条件(a)と(b)が成り立つときである．

(a) $s_1=t\Rightarrow s_1'$である$s_1'\in S$が存在するならば，

$s_2=t\Rightarrow s_2'$で$(s_1',s_2')\in R$である$s_2'\in S$が存在する．

(b) $s_2=t\Rightarrow s_2'$である$s_2'\in S$が存在するならば，

$s_1=t\Rightarrow s_1'$で$(s_1',s_2')\in R$である$s_1'\in S$が存在する． □

弱bisimulation関係を用いて，状態及びLTSの弱bisimulation等価を以下のように定義する．

【 定義4-3 】状態及びLTSの弱bisimulation等価

$Sys1=<S_1,A,T_1,s_{01}>$，$Sys2=<S_2,A,T_2,s_{02}>$ を任意のLTSとし，$S=S_1\cup S_2$とする．このとき，状態の組(s_1,s_2)を含むような弱bisimulation関係$R\subseteq S\times S$が存在するとき，状態s_1とs_2は弱bisimulation等価であるといい，

“$s_1 \sim_S s_2$”と書く．特に，初期状態の組(s_{01},s_{02})を含むような関係Rが存在するとき，$Sys1$と$Sys2$は弱bisimulation等価であるといい，“$Sys1 \sim_S Sys2$”と書く． □

また，動作式の弱bisimulation等価を次のように定める．

【 定義4-4 】動作式の弱bisimulation等価

二つの動作式B_1とB_2から得られるLTSをそれぞれ$Sys1$と$Sys2$とする．$Sys1 \sim_S Sys2$のとき，かつそのときに限りB_1とB_2は弱bisimulation等価であるといい，$B_1 \sim_S B_2$と書く． □

LTSの弱bisimulation等価に関して，任意の2つのLTSが与えられた時それらの等価性を判定するアルゴリズム[Kami 89]，及び，任意のLTSが与えられた時その最簡形を求めるアルゴリズム[Kami 90]がある．ここで，LTS Sysの最簡形とは，Sysと弱bisimulation等価なLTSの中で状態と遷移の数が最少のものをいう．以下，本論文ではLTSの最簡形を求めることをLTSの最小化と呼ぶ．

状態の弱bisimulation等価に関して，次の命題が成り立つ．

【 命題4-1 】

LTS $Sys1 = < S_1, A, T_1, s_0 >$において

$(s_1,a,s_2) \in T_1, \exists s_3 \in S_1 , s_2 \sim_S s_3 , s_2 \neq s_0$

とする．この時，以下のようにして構成されるLTSを$Sys2 = <S_2,A,T_2,s_0>$

とすると，

$$Sys1 \sim_S Sys2$$

である．ここで，$S_2=S_1$，$T_2=(T_1-\{(s_1,a,s_2)\})\cup\{(s_1,a,s_3)\}$.

すなわち，$Sys2$は，$Sys1$の遷移 $s_1 \text{—}a\rightarrow s_2$ を取り除き，そのかわりに遷移 $s_1 \text{—}a\rightarrow s_3$ を付け加えたものである．

【 証明 】

便宜上，$Sys2$を作る上で，$Sys1$の状態 s^1 に対応する$Sys2$の状態を s^2 で表す．ここで，スーパースクリプト$(1,2)$はその状態が$Sys1$のものか$Sys2$のものかを区別するのに用いている．

関係$I,R\subseteq S_1\times S_2$を次のように定める(図4-3参照).

$I=\{(s^1,s^2)\mid s^1\in S_1\}$,

$R=\{(s_p{}^1,s_q{}^2)|s_p{}^1\in S_1, s_q{}^2\in S_2, s_p{}^1\sim_S s_q{}^1,$

$\exists t_1,t_2\in(A-\{\mathrm{i}\})^*$について $s_0{}^1=t_1\Rightarrow s_1{}^1, s_1{}^1\text{—}a\rightarrow s_2{}^1, s_2{}^1=t_2\Rightarrow s_p{}^1\}$.

$Sys2$の作り方より，Iは$Sys1$と$Sys2$の初期状態の組$(s_0{}^1,s_0{}^2)$を含む．ここで，関係 $Q\subseteq S_1\times S_2$ ($\subseteq S\times S, S=S_1\cup S_2$) を $Q=I\cup R$ と定めると，Qは$Sys1$と$Sys2$の初期状態の組$(s_0{}^1,s_0{}^2)$を含む．このとき，Qが弱bisimulation関係であることを示す．

定義4-2より，$(s^1,s^2)\in Q$ と任意の $t\in(A-\{\mathrm{i}\})^*$ について次の(i), (ii)が成り立つことを示せばよい．

(i) $s^1 = t \Rightarrow s_p{}^1$ならば, $s^2 = t \Rightarrow s_q{}^2$で,

$(s_p{}^1, s_q{}^2) \in Q$ である $s_q{}^2 \in S_2$ が存在する.

(ii) $s^2 = t \Rightarrow s_q{}^2$ならば, $s^1 = t \Rightarrow s_p{}^1$で,

$(s_q{}^2, s_p{}^1) \in Q$ である $s_p{}^1 \in S_1$ が存在する.

(i)についてまず示す.

$Sys1$と$Sys2$の違いは $s_1{}^1 \text{—}a\rightarrow s_2{}^1$ と $s_1{}^2 \text{—}a\rightarrow s_3{}^2$ の違いのみであるので, 次の場合を議論すれば十分である. すなわち,

$s^1 = t \Rightarrow s_p{}^1$　　(ただし, $t = t_1 t_2 (a = \mathrm{i}), t = t_1 a t_2 (a \neq \mathrm{i})$

かつ $s^1 = t_1 \Rightarrow s_1{}^1, s_1{}^1 \text{—}a\rightarrow s_2{}^1, s_2{}^1 = t_2 \Rightarrow s_p{}^1$).

である場合を考える(図4-4(a)). 他の場合は, $Sys2$の作り方から明らかに(i)が成り立つ.

$Sys2$の作り方より, $s^2 = t_1 \Rightarrow s_1{}^2$かつ$s_1{}^2 \text{—}a\rightarrow s_3{}^2$かつ$s_3{}^2 = t_2 \Rightarrow s_q{}^2$が存在する(図4-4(b)). このとき, 命題の仮定より$s_2{}^1 \sim_S s_3{}^1$であるから$s_3{}^1 = t_2 \Rightarrow s_q{}^1$が存在して$s_p{}^1 \sim_S s_q{}^1$. よって, Rの定義より$(s_p{}^1, s_q{}^2) \in R$である. ここで$R \subseteq Q$であるから$(s_p{}^1, s_q{}^2) \in Q$. 以上より, $s^2 = t \Rightarrow s_q{}^2$が存在し, $(s_p{}^1, s_q{}^2) \in Q$である.

(ii)についても(i)と同様に示される. □

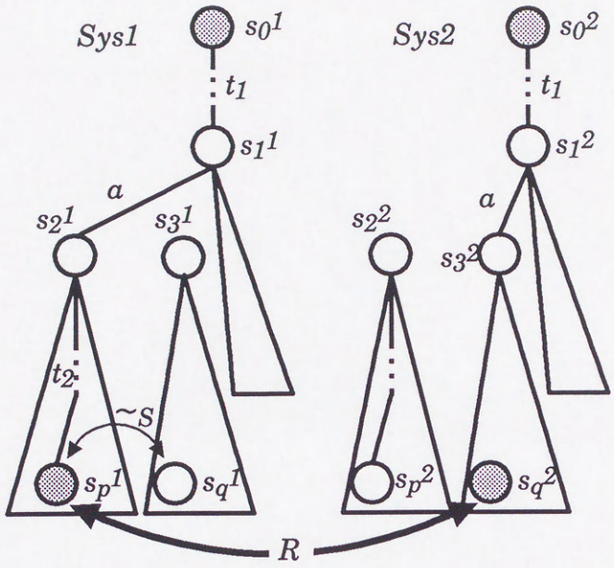


図4-3　命題4-1の説明図(1)

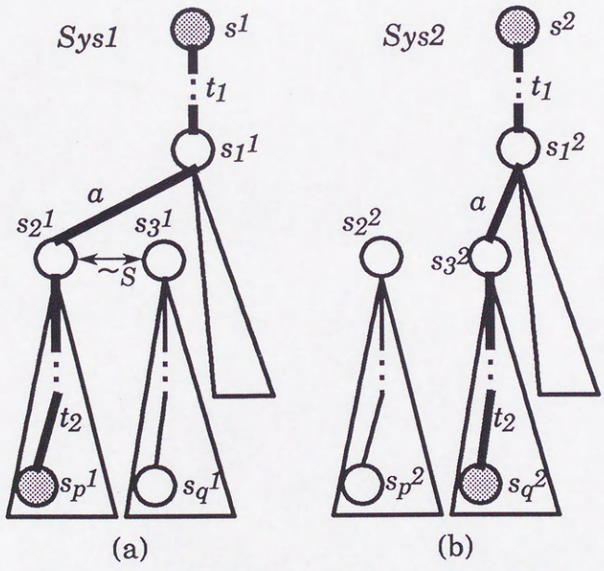


図4-4　命題4-1の説明図(2)

4.3 変数付LTS

本節では，変数付LTS(これをVTSと呼ぶ)を導入する．

以下では，まずVTSを導入し，次にその導出体系を与える．そして最後にVTSの意味を議論する．

4.3.1 VTSの導入

変数付き遷移システム(VTS)は，変数(環境によって値が決まる変数)とそれに関連した条件(選択述語およびガード)が付随したLTSであり，次の定義4-5で定める．尚，VTS $Vsys$は，与えられたLOTOSの動作式Bに対して，4.3.2に示す導出体系を用いることによって得られる．

【 定義4-5 】変数付き遷移システム(VTS)

VTS $Vsys$ は次の 7 項組である．

$Vsys = <N, G, AS, V, CA, Tr, n_0>$ ここで，

N : 状態の集合

G : ゲートの集合

AS : 抽象データ型の代数仕様　$AS = <S, OP, E>$

但し，$<S, OP>$はシグニチャ，Eは等式の集合

V : S上の変数の集合

CA : 条件アクションの集合

各条件アクション $ca \in CA$ は

$ca = action$ 又は $[cond_1]\cdots[cond_m]action$.

ここで,

$cond_i\ (1 \leqq i \leqq m)$ は $<S,OP>$ 及びV上の等式.

$action$ は次の①～③のいづれか.

① i

② $g\ d_1 \cdots d_n\ [SP_1] \cdots [SP_h]$

③ $\mathrm{i}\ (v_1 \cdots v_k\ [SP_1] \cdots [SP_h])$.

但し,

$g \in G$,

$d_j (1 \leqq j \leqq n)$ は $!t_j$ または $?x_j{:}s_j$ のどちらか

(t_j は $<S, OP>$ 及び V 上の項,

$x_j{:}s_j$ は V 中のソート s_j の変数 x_j),

$SP_j (1 \leqq j \leqq h)$ は $<S,OP>$ 及び V 上の等式,

$v_j (1 \leqq j \leqq k)$ は $?x_j{:}s_j$

Tr : 遷移の集合　　$Tr \subseteq N \times CA \times N$

n_0 : $Vsys$ の初期状態 $(n_0 \in N)$ □

$(s_1,ca,s_2) \in Tr$ (但し $ca = [cond_1]\cdots[cond_m]action$)

であるとき, これは等式"$cond_1$",…,"$cond_m$"が成立するならばアクション

"*action*"の生起により状態"s_1"から状態"s_2"への遷移が可能であることを意味し, "s_1—*ca*→s_2"と表記する. 遷移$t=(n_1,ca,n_2)\in Tr$において, 関数 $from, to: Tr \rightarrow N$, $label: Tr \rightarrow CA$ を

$$from(t) = n_1 , to(t) = n_2 , label(t) = ca$$

と定める. また, $Vsys=<N,G,AS,V,CA,Tr,n_0>$における有限な遷移の系列 $t_1 \cdots t_k (t_i \in Tr)$(但し$to(t_i) = from(t_{i+1})(i=1,\cdots,k-1)$)を"有向パス"と呼ぶ. さらに, VTSにおけるアクション系列の遷移関係—*ct*→および=*ct*⇒を定義4-1と同様に定める(ただし*ct*は条件アクションの系列).

4.3.2 VTSの導出体系の構成

BをLOTOS仕様の初期プロセス定義の動作式とする. Bに対応するVTS $Vsys$は, 次に与えるVTSの導出体系及びこれに関連した定義4-6, 定義4-7によって定める. VTSの導出体系は, 動作式から対応するVTSを導出するための公理と推論規則から成る. 尚, ここでは省略記法的なもの(**choice**式, **par**式, **let**式等)を除いたLOTOSのサブセットを扱う. 対象とするLOTOSの構文を付録Aに示す. 以下, 動作式Bにおける変数$x_1, \cdots, x_n$のオカレンスへの項$t_1, \cdots, t_n$によるネーミングクラッシュ(名前衝突)を避けた同時代入を"$[t_1/x_1, \cdots, t_n/x_n]B$"と表記する.

また, 抽象データ型の代数仕様$AS=<S, OP, E>$が与えられた時, 以下の記法を導入する[ISO 89a].

- Ω_{AS} ： ASから生成される導出体系
- $[t]_{AS}$: $<S, OP>$ 上の基礎項(ground term) t の

AS-合同類　$[t]_{AS} = \{t' \mid \Omega_{AS} \vdash t = t'\}$

さらに，推論規則の定義のため，各条件アクションについてそのゲート名を返す次のような関数$name$を導入する．

$name([cond_1]\cdots[cond_n]gv_1\cdots v_m[SP_1]\cdots[SP_h]) = g$,

$name([cond_1]\cdots[cond_n]\delta) = \delta$,

$name([cond_1]\cdots[cond_n]\mathrm{i}) = \mathrm{i}$,

$name([cond_1]\cdots[cond_n]\mathrm{i}(v_1\cdots v_m[SP_1]\cdots[SP_h])) = \mathrm{i}$.

【VTSの導出体系】

LOTOS仕様$CLS = <AS, BS>$(BS:動作仕様)が与えられたとき，導出体系Ψ_{CLS}は以下に示す公理と推論規則によって定義する．

(1) *action prefix* (internal action)

- $\mathrm{i}\ ;\mathrm{B} \xrightarrow{\ \mathrm{i}\ } \mathrm{B}$

(2) *action prefix* (communication action)

- $\mathrm{g}\,\mathrm{d}_1\cdots\mathrm{d}_n\ ;\mathrm{B} \xrightarrow{\ \mathrm{g}\,\mathrm{v}_1\cdots\mathrm{v}_n\ } [\mathrm{y}_1'/\mathrm{y}_1, \cdots, \mathrm{y}_m'/\mathrm{y}_m]\mathrm{B}$

但し，$\{\mathrm{y}_1, \cdots, \mathrm{y}_m\} = \{\mathrm{x}_i \mid \mathrm{d}_i = ?\mathrm{x}_i{:}\mathrm{s}_i, 1 \leqq i \leqq n\}$

とおき，y_j'は新しいユニークな変数名($1 \leqq j \leqq m$)とする．この時，

$v_i = ?y_j':s_i$ ($d_i = ?y_j:s_i$ $(1 \leqq i \leqq n, 1 \leqq j \leqq m)$ の時)

$v_i = !\,t_i'$ ($d_i = !t_i$ $(1 \leqq i \leqq n)$, t_i が基礎項の時.

t_i' は $[t_i]_{AS}$ の代表元)

$v_i = !\,t_i$ ($d_i = !t_i$ $(1 \leqq i \leqq n)$, t_i が基礎項でない時)

とする.

(3) *action prefix* (選択述語付き)

- $g\,d_1 \cdots d_n\,[SP_1] \cdots [SP_h]\,;B$

  $\quad -g\,v_1 \cdots v_n\,[SSP_1] \cdots [SSP_k] \rightarrow [y_1'/y_1, \cdots, y_m'/y_m]\,B$

  但し, (2) と同じ条件が成立. 更に, 次の条件が必要.

  $[y_1'/y_1, \cdots, y_m'/y_m]\,([SP_1] \cdots [SP_h])$ を $[SP'_1] \cdots [SP'_h]$ とおく. これらの選択述語 (等式) $SP'_1, \cdots, SP'_h$ の内, 基礎等式 (ground equation) を $SP''_1, \cdots, SP''_f$ とし, 残りを $SSP_1, \cdots, SSP_k$ とする $(f+k=h)$. この時, 各 SP''_i $(1 \leqq i \leqq f)$ について, $\Omega_{AS} \vdash SP''_i$.

(4) *termination*

- **exit** $-\delta \rightarrow$ **stop**

(5) *hiding*

- $$\frac{B \xrightarrow{ca} B' \quad , \quad \mathrm{name}(ca) \notin \{g_1, \cdots, g_n\}}{\textbf{hide}\ g_1, \cdots, g_n\ \textbf{in}\ B \xrightarrow{ca} \textbf{hide}\ g_1, \cdots, g_n\ \textbf{in}\ B'}$$

$B \xrightarrow{-[cond_1]\cdots[cond_k]\, g\, v_1 \cdots v_m\, [SP_1]\cdots[SP_h]} B'$,
$name(ca) \in \{g_1, \cdots, g_n\}$

- ---

hide $g_1, \cdots, g_n$ **in** $B \xrightarrow{-[cond_1]\cdots[cond_k]\, i\, (t_1 \cdots t_f\, [SP_1]\cdots[SP_h])}$ **hide** $g_1, \cdots, g_n$ **in** B'

但し, $v_1, \cdots, v_m$ の内, ?x:s の形 (すなわち, 変数宣言の形) をしているものを $t_1, \cdots, t_f$ とする.

(6) *enabling*

- $\dfrac{B1 \xrightarrow{-ca} B1' \quad , \quad name(ca) \neq \delta}{B1 >> B2 \xrightarrow{-ca} B1' >> B2}$

- $\dfrac{B1 \xrightarrow{-[cond_1]\cdots[cond_n]\,\delta} B1'}{B1 >> B2 \xrightarrow{-[cond_1]\cdots[cond_n]\,i} B2}$

(7) *disabling*

- $\dfrac{B1 \xrightarrow{-ca} B1' \quad , \quad name(ca) \neq \delta}{B1\,[>\,B2 \xrightarrow{-ca} B1'\,[>\,B2}$

- $\dfrac{B1 \xrightarrow{-[cond_1]\cdots[cond_n]\,\delta} B1'}{B1\,[>\,B2 \xrightarrow{-[cond_1]\cdots[cond_n]\,\delta} B1'}$

- $\dfrac{B2 \xrightarrow{-ca} B2'}{B1\,[>\,B2 \xrightarrow{-ca} B2'}$

(8) *choice*

- $$\frac{B1 \xrightarrow{ca} B1'}{B1 \,[]\, B2 \xrightarrow{ca} B1'}$$

- $$\frac{B2 \xrightarrow{ca} B2'}{B1 \,[]\, B2 \xrightarrow{ca} B2'}$$

(9) *guarding*

- $$\frac{B \xrightarrow{ca} B' \quad , \quad P\text{が基礎等式であり}, \Omega_{AS} \vdash P}{[P] -> B \xrightarrow{ca} B'}$$

- $$\frac{B \xrightarrow{ca} B' \quad , \quad P\text{が基礎等式でない}}{[P] -> B \xrightarrow{[P]ca} B'}$$

(10) *process instantiation*

- $$\frac{[t_1/x_1, \cdots, t_m/x_m]\, Bp\, [g_1/h_1, \cdots, g_n/h_n] \xrightarrow{ca} B'}{p\,[g_1, \cdots, g_n]\,(t_1, \cdots, t_m) \xrightarrow{ca} B'}$$

但し，Bp は p のプロセス抽象.

プロセス p の仮ゲートパラメータは

$h_1, \cdots, h_n$

であり，仮値パラメータは

$x_1, \cdots, x_m$

である.

(11) *relabelling*

$$\bullet\ \frac{B \ -ca\rightarrow\ B'}{B\,[g_1/h_1, \cdots, g_n/h_n] \ -ca'\rightarrow\ B'\,[g_1/h_1, \cdots, g_n/h_n]}$$

但し, $ca = [cond_1] \cdots [cond_k]\, g\, v_1 \cdots v_m\, [SP_1] \cdots [SP_l]$

$ca' = [cond_1] \cdots [cond_k]\, g\, v_1 \cdots v_m\, [SP_1] \cdots [SP_l]$

($g \notin \{h_1, \cdots, h_n\}$ の時)

$ca' = [cond_1] \cdots [cond_k]\, g_i\, v_1 \cdots v_m\, [SP_1] \cdots [SP_l]$

($g = h_i$ $(1 \leqq i \leqq n)$ の時)

(12) *general parallel*

$$\bullet\ \frac{B1 \ -ca\rightarrow\ B1' \quad , \quad name(ca) \notin G \cup \{\delta\}}{B1|[G]|\,B2 \ -ca\rightarrow\ B1'|[G]|\,B2}$$

$$\bullet\ \frac{B2 \ -ca\rightarrow\ B2' \quad , \quad name(ca) \notin G \cup \{\delta\}}{B1|[G]|\,B2 \ -ca\rightarrow\ B1|[G]|\,B2'}$$

$$\bullet\ \frac{B1 \ -[cond_{11}] \cdots [cond_{1r}]\, \delta\rightarrow\ B1' \,,\ B2 \ -[cond_{21}] \cdots [cond_{2s}]\, \delta\rightarrow\ B2'}{B1|[G]|\,B2 \ -[cond_{11}] \cdots [cond_{1r}]\, [cond_{21}] \cdots [cond_{2s}]\, \delta\rightarrow\ B1'|[G]|\,B2'}$$

$$\bullet\ \frac{\begin{array}{l} B1 \ -[cond_{11}] \cdots [cond_{1r}]\, g\, v_{11} \cdots v_{1n}\, [SP_{11}] \cdots [SP_{1k}] \rightarrow\ B1' \quad , \\ B2 \ -[cond_{21}] \cdots [cond_{2s}]\, g\, v_{21} \cdots v_{2n}\, [SP_{21}] \cdots [SP_{2h}] \rightarrow\ B2' \quad , \ g \in G \end{array}}{\begin{array}{l} B1|[G]|\,B2 \\ -[cond_{11}] \cdots [cond_{1r}]\, [cond_{21}] \cdots [cond_{2s}]\, g\, v_1 \cdots v_n\, [SP_1] \cdots [SP_m] \rightarrow \\ \qquad\qquad [\sigma 1]\,B1'|[G]|\,[\sigma 2]\,B2' \end{array}}$$

但し, 各 v_{1i}, v_{2i} $(1 \leqq i \leqq n)$ は次の内のいずれかの形.

(a) $v_{1i} = ?\,x_{1i} : s_i$, $v_{2i} = ?\,x_{2i} : s_i$

(b) $v_{1i} = ?\,x_{1i} : s_i$, $v_{2i} = !$ ソート s_i の項 t_{2i}

(c) $v_{1i} = !$ ソート s_i の項 t_{1i} , $v_{2i} = ?\,x_{2i} : s_i$

(d) v_{1i} = ! ソート s_i の基礎項 t_{1i} ， v_{2i} = ! ソート s_i の基礎項 t_{2i}

($t_{1i} = t_{2i}$ が基礎等式で， $\Omega_{AS} \vdash t_{1i} = t_{2i}$)

(e) v_{1i} = ! ソート s_i の項 t_{1i} ， v_{2i} = ! ソート s_i の項 t_{2i}

($t_{1i} = t_{2i}$ が基礎等式でない)

(a) の時， $v_i = ?x_i : s_i$ (x_i は新しいユニークな変数名)，

$\sigma_{1i} = x_i / x_{1i}$ ，

$\sigma_{2i} = x_i / x_{2i}$.

(b) の時， $v_i = !t_{2i}$ ，

$\sigma_{1i} = t_{2i} / x_{1i}$ ，

σ_{2i} = 恒等代入 .

(c) の時， $v_i = !t_{1i}$ ，

σ_{1i} = 恒等代入 ，

$\sigma_{2i} = t_{1i} / x_{2i}$.

(d) の時， $v_i = !t_{1i}$ ，

σ_{1i} = 恒等代入 ，

σ_{2i} = 恒等代入 .

(e) の時， $v_i = !t_{1i}$ ，

σ_{1i} = 恒等代入 ，

σ_{2i} = 恒等代入 ，

$[=(t_{1i}, t_{2i})] \in ([SP_1] \cdots [SP_m])$.

また， $[\sigma 1] = [\sigma_{11}, \cdots, \sigma_{1n}]$ ，

$[\sigma 2] = [\sigma_{21}, \cdots, \sigma_{2n}]$

とおき，

$[\sigma 1]([SP_{1j}]) \in \{[SP_1], \cdots, [SP_m]\}$ (各 SP_{1j} $(1 \leqq j \leqq k)$ について)

$[\sigma 2]([SP_{2j}]) \in \{[SP_1], \cdots, [SP_m]\}$ (各 SP_{2j} $(1 \leqq j \leqq h)$ について)

とする.

(13) *interleaving*

$$\bullet\ \frac{B1\ |[\ ]|\ B2\ \xrightarrow{-ca\rightarrow}\ B'}{B1\ |||\ B2\ \xrightarrow{-ca\rightarrow}\ B'}$$

(14) *full synchronization*

$$\bullet\ \frac{B1\ |[\ g_1, \cdots, g_n\ ]|\ B2\ \xrightarrow{-ca\rightarrow}\ B'}{B1\ ||\ B2\ \xrightarrow{-ca\rightarrow}\ B'}$$

但し，$g_1, \cdots, g_n$はすべてのゲートのリスト． □

【 定義4-6 】動作式のderivatives

LOTOS仕様$CLS = <AS,BS>$が与えられたとき，これに関連した動作式Bのderivativeの集合$DER_{CLS}(B)$は，以下の条件を満たす最も小さな集合である．

① $B \in DER_{CLS}(B)$

② $B' \in DER_{CLS}(B)$で，ある条件アクションcaについて$\Psi_{CLS} \vdash B'\text{—}ca\rightarrow B''$ならば$B'' \in DER_{CLS}(B)$．ここで"$\Psi_{CLS} \vdash B'\text{—}ca\rightarrow B''$"はVTSの導出体系$\Psi_{CLS}$により遷移$B'\text{—}ca\rightarrow B''$が導出されることを表す． □

動作式Bのderivativesの導出の上で出現する条件アクションの集合を$CA_{CLS}(B)$，条件アクション中に出現する変数の集合を$V_{CLS}(B)$，同様にゲートの集合を$G_{CLS}(B)$とする．

【 定義4-7 】動作式に対応するVTS

LOTOS仕様$CLS=<AS,BS>$の初期プロセス定義の動作式Bに対応するVTS $VTS_{CLS}(B)$は，次の7項組である．

$VTS_{CLS}(B)=<N,G,AS,V,CA,Tr,B>$ ここで，

$N=DER_{CLS}(B)$, $G=G_{CLS}(B)$, $V=V_{CLS}(B)$,

$CA=CA_{CLS}(B)$, $Tr=T_{CLS}$. ただし，

$T_{CLS}=\{(B_1,ca,B_2)|B_1,B_2\in DER_{CLS}(B),ca\in CA_{CLS}(B),\Psi_{CLS}\vdash B_1\text{—}ca\rightarrow B_2\}$

□

以上の様にして，動作式が与えられたとき，それに対応するVTSを得ることができる．この時，VTS上に現れる変数はユニーク(グローバル)であることに注意されたい．

4.3.3 VTSの意味

LOTOS仕様$CLS=<AS,BS>$の初期プロセス定義の動作式Bに対応するVTS $VTS_{CLS}(B)$の意味は，$VTS_{CLS}(B)$に出現する変数に具体値を割当て，関連した条件(ガードと選択述語)をASの下で解釈すること(具現化)によって得られるLTSとして定義される．

抽象データ型の代数仕様$AS=<S,OP,E>$及びS上の変数の集合$V=\{x_1,\cdots,x_n\}$が与えられたとする．この時，以下の記法を導入する．

● $Q_{s,AS}$： ソート$s \in S$の基礎項のAS-合同類の集合(つまりsの値領域) $Q_{s,AS}=\{[t]_{AS} | t$は$s \in S$の基礎項$\}$

● $sort(x_i)$： 変数$x_i \in V$のソート

● $A_{V,AS}$： Vの変数の組から成る集合$\{(x_1,\cdots,x_n)\}$から変数の値領域$Q_{sort(x_1),AS} \times \cdots \times Q_{sort(x_n),AS}$へのすべての関数の集合(つまり変数への値割り当て)

● $\sigma(r)$： 項または等式をrとした時，r中に現れる変数を$\sigma \in A_{V,AS}$による値を示す基礎項(合同類の代表元)で置き換えて得られる新たな項または等式

● $x \leftarrow v$： 変数$x \in V$への値$v \in Q_{sort(x),AS}$の割当て

● $\sigma[y_1 \leftarrow v_1,\cdots,y_m \leftarrow v_m](\sigma[y_1 \leftarrow v_1,\cdots,y_m \leftarrow v_m] \in A_{V,AS})$

： $\sigma \in A_{V,AS}$に対して，$y_1 \leftarrow v_1,\cdots,y_m \leftarrow v_m$だけが$\sigma$と異なる値割当て

VTSの意味は，以下に与える具現化規則及びこれに関連した定義4-8～定義4-10によって定める．

【具現化規則】

VTS $Vsys = <N,G,AS,V,CA,Tr,n_0>$ が与えられると，次の①～③により遷移集合$RT(Vsys)$を得る．この①～③を具現化規則と呼ぶ．

① $(q, [cond_1] \cdots [cond_m]\,\mathbf{i}, q') \in Tr$,

$\exists\sigma$ について $\Omega_{AS} \vdash \sigma(cond_j)\ (1 \leqq j \leqq m)$.

$\Rightarrow ((q, \sigma), \mathbf{i}, (q', \sigma)) \in RT(Vsys)$.

② $(q, [cond_1] \cdots [cond_m] g d_1 \cdots d_n [SP_1] \cdots [SP_h], q') \in Tr$,

$\exists\sigma$ について $\Omega_{AS} \vdash \sigma(cond_j)\ (1 \leqq j \leqq m)$,

$\Omega_{AS} \vdash \sigma_{[y_1 \leftarrow u_1, \cdots, y_k \leftarrow u_k]}(SP_w)\ (1 \leqq w \leqq h)$.

ここで, $\{y_1, \cdots, y_k\} = \{x_i \mid d_i = ?x_i{:}s_i, 1 \leqq i \leqq n\}$,

$u_z \in Q_{sort(y_z),AS}\ (1 \leqq z \leqq k)$.

$\Rightarrow ((q, \sigma), g\langle v_1 \cdots v_n \rangle, (q', \sigma_{[y_1 \leftarrow u_1, \cdots, y_k \leftarrow u_k]}))$

$\in RT(Vsys)$. ここで,

$v_i = [\sigma(t_i)]_{AS}$($d_i = !\,t_i$ $(1 \leqq i \leqq n)$ の時)

$v_i = u_z$ $(d_i = ?x_i{:}s_i, x_i = y_z\ (1 \leqq i \leqq n, 1 \leqq z \leqq k)$ の時)

③ $(q, [cond_1] \cdots [cond_m]\,\mathbf{i}(?y_1{:}s_1 \cdots ?y_k{:}s_k\,[SP_1] \cdots [SP_h]), q') \in Tr$,

$\exists\sigma$ について $\Omega_{AS} \vdash \sigma(cond_j)\ (1 \leqq j \leqq m)$,

$\Omega_{AS} \vdash \sigma_{[y_1 \leftarrow u_1, \cdots, y_k \leftarrow u_k]}(SP_w)\ (1 \leqq w \leqq h)$.

ここで, $u_z \in Q_{s_z,AS}\ (1 \leqq z \leqq k)$.

$\Rightarrow ((q, \sigma), \mathbf{i}, (q', \sigma_{[y_1 \leftarrow u_1, \cdots, y_k \leftarrow u_k]})) \in RT(Vsys)$. □

$RT(Vsys)$に含まれる状態の集合を$States(RT(Vsys))$ またアクションの集合を$Actions(RT(Vsys))$と表す.

【 定義4-8 】初期値割当てによる再構成LTS

初期値割当て$\iota \in A_{V,AS}$及びVTS $Vsys = <N,G,AS,V,CA,Tr,n_0>$が与えられたとき，次のLTS $RTS_\iota(Vsys)$ を$Vsys$の再構成LTSと呼ぶ．

$$RTS_\iota(Vsys) = <States(RT(Vsys)),$$
$$Actions(RT(Vsys)),RT(Vsys),(n_0,\iota)> \quad \square$$

【定義4-9】初期値割当てによるVTSの再構成遷移集合

初期値割当て$\iota \in A_{V,AS}$によるVTS $Vsys = <N,G,AS,V,CA,Tr,n_0>$の再構成遷移集合$RT_\iota(Vsys)$は，$Vsys$の再構成遷移集合$RT(Vsys)$から，状態$(n_0, \iota)$より到達不能な遷移を取り除いて得られる集合である． □

【 定義4-10 】初期値割当てιによるVTSの意味

初期値割当て$\iota \in A_{V,AS}$によるVTS $Vsys = <N,G,AS,V,CA,Tr,n_0>$の意味は，以下で定義されるLTS $TS_\iota(Vsys)$である．

$$TS_\iota(Vsys) = <States(RT_\iota(Vsys)),$$
$$Actions(RT_\iota(Vsys)), RT_\iota(Vsys), (n_0, \iota)> \quad \square$$

定義4-8及び定義4-10より明らかに次の命題が成り立つ．

【 命題4-2 】

VTS $Vsys1$, $Vsys2$ が与えられた時，次が成り立つ．

$$RTS_\iota(Vsys1) \sim_S RTS_\iota(Vsys2)$$

$$\Rightarrow TS_{\iota}(Vsys1) \sim_S TS_{\iota}(Vsys2)$$

ここで，ιは任意の初期値割当て．　□

次に，LOTOS仕様$CLS = <AS, BS>$の初期プロセス定義の動作式Bに対応して得られるVTS $VTS_{CLS}(B)$の意味であるLTS $TS_{\iota}(VTS_{CLS}(B))$と，Bの意味を直接与えるLTS $TS_{CLS}(B)$との関係について調べる．ここで，$TS_{CLS}(B)$は[ISO 89a]で定められた体系によりBに対応して得られるLTSである．

まず，2つのLTSの強bisimulation等価を[ISO 89a]に基づき次の様に定める．すなわち，強bisimulation関係は定義4-2において$=t\Rightarrow$の代わりに$-a\rightarrow$で置き換えて定義され，さらに強bisimulation等価は強bisimulation関係に基づいて定義4-3と同様に定められる．この定義より，強bisimulation関係は弱bisimulation関係でもある．

LOTOS勧告書[ISO 89a]で示されるLTSの導出体系(以下体系Aと呼ぶ)と上で示したVTSの導出体系(以下体系Bと呼ぶ)の違いは，変数の取り扱い方による．すなわち，体系Bにおいては変数をそのままVTSの上に残すが，変数を具体値で具現化する際に体系Aで導出される遷移と対応付けが行なえるように遷移を導出する体系となっている．従って，次の命題が成り立つ．

【 命題4-3 】LTSの導出体系とVTSの導出体系の関係

LOTOS仕様CLS＝<AS,BS>に関連した閉じた動作式をBとする．この時

Sys_1とSys_2は強bisimulation等価である．ここで，

$Sys_1 = TS_{CLS}(B)$, $Sys_2 = TS_{\iota}(VTS_{CLS}(B))$

(ιは任意の初期値割当て). □

次の例4-1はこの関係を例示している．これは両体系の違いを示す典型的な例となっている．

【 例 4-1 】強bisimulation 等価の例

次のような動作式Bを考える(関連したデータ型の代数仕様については省略，ガードおよび選択述語の書き方は省略形を採用)．

B = **hide** g **in** $g\,?\,x : nat\,[0 \leqq x \leqq 1]\,;\, f\,?\,y : nat\,;$

($[0 \leqq y \leqq 1] \,->\, (h\,!\,x\,;$ **stop** $|[h]|\, h\,!\,y\,;$ **stop**)

[] $[y \geqq 2] \,->\, e\,!\,err\,;$ **stop**)

Bに対応するLTS $TS_{CLS}(B)$は図4-5の通りである．また，Bに対応するVTS $VTS_{CLS}(B)$は図4-6の通りである．さらに，このVTSの意味は図4-7のLTS $TS_{\iota}(VTS_{CLS}(B))$である．ここで，次のような状態間の関係を$TS_{CLS}(B)$と$TS_{\iota}(VTS_{CLS}(B))$の間にとることができるので，両者は強bisimulation等価であることがわかる．

$s0 \Leftrightarrow (s0', \iota)$, $s1 \Leftrightarrow (s1', \sigma 1)$, $s2 \Leftrightarrow (s1', \sigma 5)$, $s3 \Leftrightarrow (s2', \sigma 2)$, $s4 \Leftrightarrow (s2', \sigma 3)$, $s5 \Leftrightarrow (s2', \sigma 4)$, $s6 \Leftrightarrow (s2', \sigma 6)$, $s7 \Leftrightarrow (s2', \sigma 7)$, $s8 \Leftrightarrow (s2', \sigma 8)$, $s9 \Leftrightarrow (s3', \sigma 2)$, $s9 \Leftrightarrow (s3', \sigma 7)$, $s10 \Leftrightarrow (s4', \sigma 4)$, $s10 \Leftrightarrow (s4', \sigma 8)$, … □

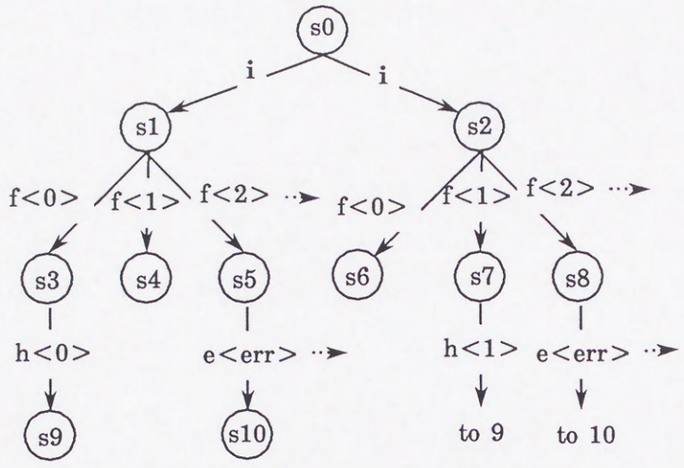


図4-5　LTS $TS_{CLS}(B)$: LTS例

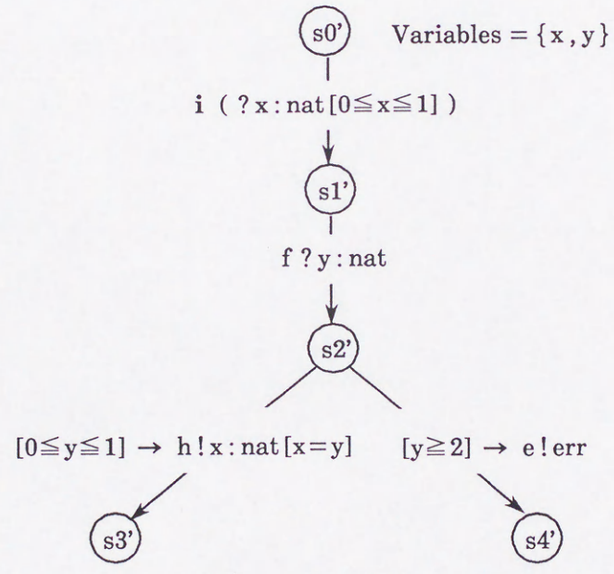
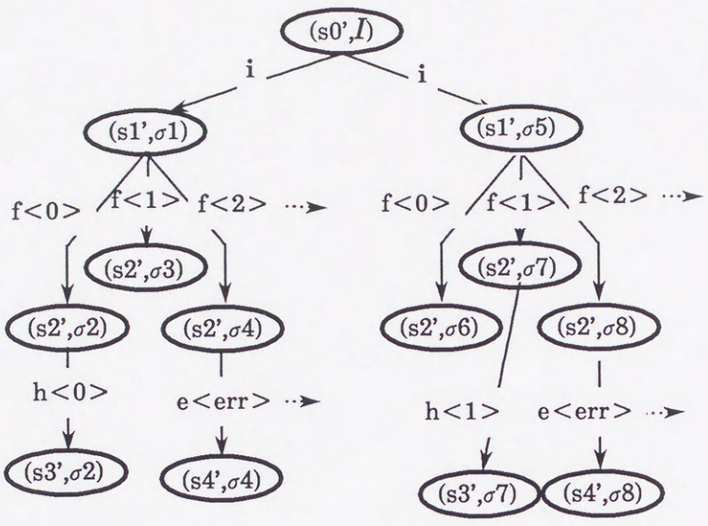


図4-6　VTS $VTS_{CLS}(B)$: VTS例

以上のように，具現化によってVTSの意味を与えることができる．これに基づいて，VTSの等価性を次のように定める．

【 定義4-11 】VTSの等価性

初期値割当てιが与えられたとき，2つのVTS $Vsys_1$, $Vsys_2$は，それぞれの意味を与えるLTS $Sys_1 = TS_\iota(Vsys_1)$, $Sys_2 = TS_\iota(Vsys_2)$ において$Sys_1 \sim_S$



where,
I: a random assignment to x and y

$\sigma 1 = I_{[x \leftarrow 0]}$ $\sigma 2 = \sigma 1_{[y \leftarrow 0]}$ $\sigma 3 = \sigma 1_{[y \leftarrow 1]}$
$\sigma 4 = \sigma 1_{[y \leftarrow 2]}$ $\sigma 5 = I_{[x \leftarrow 1]}$ $\sigma 6 = \sigma 5_{[y \leftarrow 0]}$
$\sigma 7 = \sigma 5_{[y \leftarrow 1]}$ $\sigma 8 = \sigma 5_{[y \leftarrow 2]}$

図4-7　$TS_I(VTS_{CLS}(B))$: VTSの意味例

Sys_2が成り立つ時，その時に限ってιのもとで等価であるという．ιが明らかな時は，単に$Vsys_1$と$Vsys_2$は等価であるという．　□

以下の議論においては，VTS及びVTSの意味であるLTSが有限である場合を対象とする．ここで有限であるとは，状態，遷移，変数が有限であることを言う．

4.3.4　木状VTS

試験スイートとしての性質を考慮し，標準的な構造として木状構造をもつVTSを考える．一般に，4.3で定めた導出体系により得られるVTSは，次のように定める合流が含まれている．

【 定義4-12 】合流

VTS $Vsys=<N,G,AS,V,CA,Tr,n_0>$ において，2つの遷移$t=(n_1,ca,n_2)$, $t'=(n_1',ca',n_2)\in Tr$(但し$n_1\neq n_1'$ and/or $ca\neq ca'$)が存在するとき，この2つの遷移t,t'は合流であるという． □

合流である遷移の中で，ループと呼ぶ特別な性質のものを次のように定める．

【 定義4-13 】ループ

VTS $Vsys=<N,G,AS,V,CA,Tr,n_0>$ において，2つの遷移$t=(n_1,ca,n_2)$, $t'=(n_1',ca',n_2)\in Tr$が合流であるものとする．このとき，ある$k,h(k<h)$について$t_k=t$, $t_h=t'$なる有向パス$l=t_1\cdots t_k\cdots t_h$が存在するとき，この合流はループであるという． □

VTSにおいて，ループ以外の合流を含まない時，ループによる合流を開けば(unfold)，木状の構造と見なすことができる．例えば，後で示す図4-11(a)において状態3から1へのループは木状の構造になっている．そこで，このようなVTSを木状VTSと呼ぶ．明らかに，一般のループ以外の合流を含むVTSはその合流を開くことにより木状VTSに変換することができる．さらに，次のようにして，この変換において合流を開く際に変数名をユニークにすることができる．すなわち，図4-8(a)のような合流が存在した場合，これを単純に開くと同図(b)のようになる．このとき，$B=B_1=B_2$である．今，変数xがB中に

含まれていたとする．このとき，当然xはB_1及びB_2中に含まれる．ここで，B_1(あるいはB_2)中の全てのxの出現をx'で置き換えても全体の意味が変わらないならば，B_2(あるいはB_1)中の全てのxの出現をx"で置き換えても全体の意味は変わらない．但し，x', x"は他で出現しない，互いに異なる新たな変数名であるものとする．このようにして，B_1, B_2中の置き換え可能な変数を全て他で出現しない変数名で置き換えることで，全体として変数名をユニークにすることができる．

そこで，以下では対象とするVTSとしては変数がユニークな木状VTSのみを考え，これを単にVTSと呼ぶ．

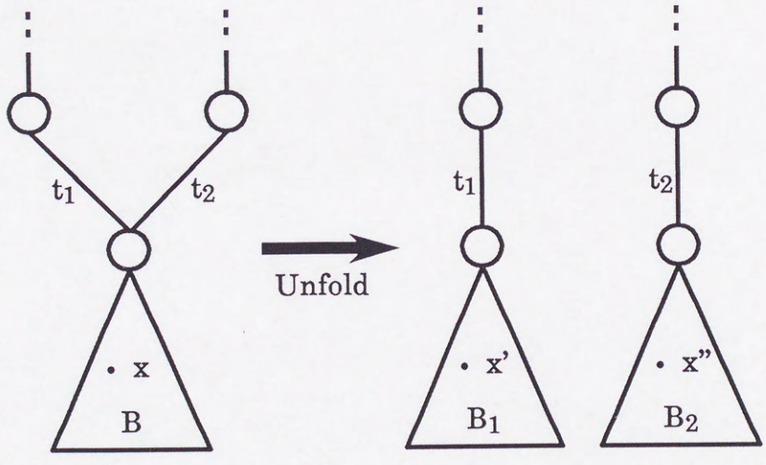


図4-8　合流の展開

4.4 LOTOS仕様からの試験系列の生成法

本節では，前節で導入したVTSを用いた，LOTOS仕様より試験系列を生成する形式的な手法(最小木法)を提案する．まず，VTSの等価性を明らかにすることによって冗長な状態の無い形(これを既約VTSと呼ぶ)を定める．これに基づき，本手法では，与えられた仕様からVTSを導出し，次にこれを既約VTSに変形し，これに基づいて試験スイートを生成する．

4.4.1 最小木法の構成

最小木法は図4-9のように構成される．

まず，4.3.2で与えたVTSの導出体系によって，与えられたLOTOS仕様からVTSを得る．さらにLTSへの変換規則である具現化規則によって対応するLTSを得る．次にその具現化されたLTSの最小化を行い，その際の情報を基にVTSの簡約化を行うことによって既約VTSを得る．さらに最小TreeLOTOSへの変換規則によって，最小TreeLOTOSとして試験スイートを得る．

4.4.2 既約VTSの導出

本小節では，上で述べた最小木法の構成に基づき，既約VTSの導出法を与える．

VTSにおける状態の等価性を議論するために，変数に関する性質を明らか

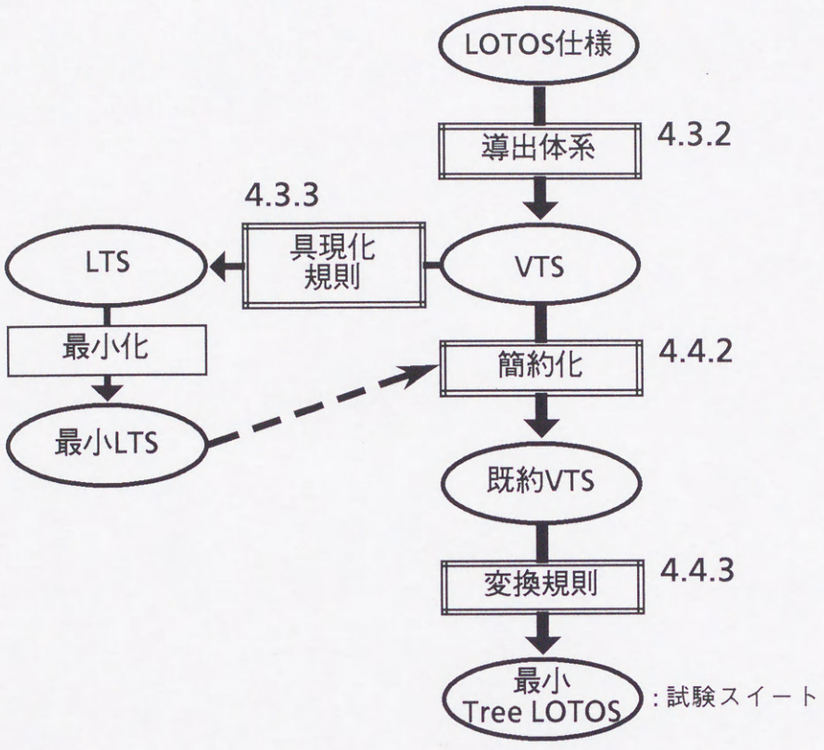


図4-9　既約VTSを用いた試験スイートの導出の概要

にする必要がある．そのために，ここでは“独立変数”及び“変数既約形(variable-irreducible form)”の概念を導入する．まず独立変数を次のように定める．

尚，以下では具現化する際の任意の初期値割当てιが与えられているものとして議論を進める．

【 定義4-14 】独立変数

VTS $Vsys=<N,G,AS,V,CA,Tr,n_0>$において，$n\in N$に対して$Ind(n)\subseteq V$を以下の条件を満たす最小の集合と定める．

定義4-8により定まる$Vsys$の再構成LTS $RTS_\iota(Vsys)$を$Sys=<S,A,T,s_0>$

とおく．$x \in V$が与えられた時，

$\forall \sigma \in A_{V,AS}, \forall v_1, v_2 \in Q_{sort(x),AS} (v_1 \neq v_2)$に対して，

$(n, \sigma[x \leftarrow v_1]) \sim_S (n, \sigma[x \leftarrow v_2])$

但し，$(n, \sigma[x \leftarrow v_1]), (n, \sigma[x \leftarrow v_2]) \in S$.

$\Rightarrow$ $x \in Ind(n)$. □

定義4-14に基づき，関係“$\sim_{(n_i, n_j)}$”を次のように定める．

【 定義4-15 】値割当て間の関係“$\sim_{(n_i, n_j)}$”

VTS $Vsys = <N, G, AS, V, CA, Tr, n_0>$ の意味として得られるLTS $TS_t(Vsys)$を

$Sys_2 = <S, A, T, s_0>$

とする．また，$n_i, n_j \in N$, $\sigma, \sigma' \in A_{V,AS}$が与えられたとする．$V_i, V_j \subseteq V$を $V_i = V - Ind(n_i)$, $V_j = V - Ind(n_j)$ とおき，次の条件(1),(2)が成り立つとき，そのときに限ってσとσ'はn_i, n_jについて等価であるといい，$\sigma \sim_{(n_i, n_j)} \sigma'$と表す．

(1)全ての$x \in V_i$についてσとσ'におけるxの値が等しい．

(2)全ての$x \in V_j$についてσ'とσにおけるxの値が等しい． □

定義4-15より明らかに次の命題が成り立つ．

【 命題4-4 】

関係“$\sim_{(n_i, n_j)}$”について，次の(1)から(3)が成り立つ．

(1) $\sigma \sim_{(n_i, n_j)} \sigma$.

(2) $Ind(n_i)=Ind(n_j)$のとき

$\sigma 1 \sim_{(n_i,n_j)} \sigma 2 \Rightarrow \sigma 2 \sim_{(n_i,n_j)} \sigma 1$.

(3) $Ind(n_h)=Ind(n_i)=Ind(n_j)$のとき

$\sigma 1 \sim_{(n_h, n_i)} \sigma 2$, $\sigma 2 \sim_{(n_i, n_j)} \sigma 3 \Rightarrow \sigma 1 \sim_{(n_h, n_j)} \sigma 3$. □

次に，変数既約形を定めるために，置き換え可能変数を次のように定める．

【 定義4-16 】置き換え可能変数

VTS $Vsys1=<N,G,AS,V,CA,Tr,n_0>$において，出現する変数$x \in V$を$y \in V(y \neq x)$で置き換えて得られるVTS $Vsys2=<N,G,AS,V',CA,Tr,n_0>$(但し$V'=V-\{x\}$)が$Vsys1$と等価であるとき，そのときに限って$Vsys1$における変数$x$は$y$に置き換え可能であるといい，また$x$は置き換え可能変数であるという． □

VTS $Vsys=<N,G,AS,V,CA,Tr,n_0>$において，置き換え可能変数が存在しないとき，$Vsys$は変数既約形であるという．ここでは，任意のVTSに対してそれと等価な変数既約形が存在するものと仮定し，以下ではその変数既約形を議論の対象とする．

以上の準備のもとで，VTSの状態間の関係"$\sim_N$"を，LTS上での状態の等価性を表す弱bisimulation関係"$\sim_S$"を用いて次のように定める．

【 定義4-17 】VTSの状態間の関係"$\sim_N$"

VTS　$Vsys=<N,G,AS,V,CA,Tr,n_0>$の意味として得られるLTS TS_t ($Vsys$)を

$Sys_2 = < S , A , T , s_0 >$

とする．n_i, n_jが与えられたとき，状態集合Sの部分集合$B_i = \{(n_i, \sigma) | (n_i, \sigma) \in S\}$, $B_j = \{(n_j, \sigma) | (n_j, \sigma) \in S\}$に対して，次の(1)から(3)が成立する時，その時に限って状態集合Nの要素n_iとn_jが“等価である”といい，“$n_i \sim_N n_j$”と表す．

(1) $Ind(n_i)=Ind(n_j)$

(2) $s_i = (n_i, \sigma_1) \in B_i \Rightarrow \exists s_j = (n_j, \sigma_1') \in B_j$,

$s_i \sim_S s_j , \sigma_1 \sim_{(n_i, n_j)} \sigma_1'$

(3) $s_j = (n_j, \sigma_2) \in B_j \Rightarrow \exists s_i = (n_i, \sigma_2') \in B_i$,

$s_j \sim_S s_i , \sigma_2 \sim_{(n_j, n_i)} \sigma_2'$. □

上のようにして定められた関係“$\sim_N$”について，次の命題が成り立つ．

【 命題4-5 】

関係“$\sim_N$”はVTSの状態集合N上の同値関係である．

【 証明 】

定義4-17及び命題4-4(1)より，反射律が成り立つ．また，定義4-17及び命題4-4(2)より，対称律が成り立つ．さらに，定義4-17及び命題4-4(3), 関係“$\sim_S$”の推移性から，推移律が成り立つ． □

同値関係“$\sim_N$”に基づいて，次の定理が成り立つ．

【 定理4-1 】

VTS $Vsys1=<N_1,G,AS,V,CA_1,Tr_1,n_0>$において，有向パス$t_1\cdots t_k(k\geqq 1)$(但し$from(t_1)=n_0$)が存在し，ある$p,q(1\leqq q\leqq p<k)$について$from(t_q)\sim_N to(t_p)$であるとき，$Vsys2=<N_2,G,AS,V,CA_2,Tr_2,n_0>$は$Vsys1$と等価である．ここで，

$Tr_2=(Tr_1-\{t_i|i=p,\cdots,k\})\cup\{t_p'\}$

(但しt_p'は$from(t_p')=from(t_p),label(t_p')=label(t_p),$

$to(t_p')=from(t_q)$なる遷移)

$N_2=\{n_0\}\cup\{to(t)|t\in Tr_2\}$, $CA_2=\{label(t)|t\in Tr_2\}$.

【 証明 】

VTS $Vsys1=<N_1,G,AS,V,CA_1,Tr_1,n_0>$及び$Vsys2=<N_2,G,AS,V,CA_2,Tr_2,n_0>$の意味として得られるLTS $TS_\iota(Vsys1)$, $TS_\iota(Vsys2)$をそれぞれ$Sys1$, $Sys2$とする．また，$Vsys1$及び$Vsys2$の再構成LTS $RTS_\iota(Vsys1)$, $RTS_\iota(Vsys2)$をそれぞれ$Sys_01=<S_1,A_1,T_1,s_0>$, $Sys_02=<S_2,A_2,T_2,s_0>$とする．

定義4-17より，$Vsys1$の遷移$t_p=(from(t_p),ca,to(t_p))\in Tr_1$に4.3で与えた具現化規則を適用して得られる$Sys_01$の遷移のうちの任意の一つを

$((from(t_p),\sigma_1),a,(to(t_p),\sigma_2))\in T_1$

と表すと，これに一対一に対応して，Sys_02に，$Vsys2$の遷移$t_p'=(from(t_p'),ca,to(t_p'))\in Tr_2$に具現化規則を適用して得られる

$((from(t_p'), \sigma_1), a, (to(t_p'), \sigma_2')) \in T_2$, (ただし, $\sigma_2 \sim_{(to(t_p), to(t_p'))} \sigma_2'$

なる遷移が存在し, $(to(t_p), \sigma_2) \sim_S (to(t_p'), \sigma_2')$である. このとき, 命題4-1より

$Sys_01 \sim_S Sys_01'$

(ただし, $Sys_01' = <S_1, A_1, T_1', s_0>$, $T_1' = (T_1 - \{((from(t_p), \sigma_1), a, (to(t_p), \sigma_2))\}) \cup \{((from(t_p'), \sigma_1), a, (to(t_p'), \sigma_2))\}$)

である. t_pを具現化して得られる全てのSys_01の遷移についてこれを繰り返して得られるLTSを$Sys_01^{(n)}$と表すと, $Sys1_0^{(n)} = Sys_02$である. よって, $Sys_01 \sim_S Sys_02$が成り立つ. ここで, 命題4-2より$Sys1 \sim_S Sys2$である.

これより, 定義4-11から$Vsys2$は$Vsys1$と等価である. □

定理4-1は, 等価である状態を同一視することによってVTSをそのパスについて簡約化(reduction)することができることを示している(図4-10). このとき, 簡約化が適用できる有向パスを“簡約化可能パス(reducible path)”と呼ぶ.

パスの簡約化に関する定理4-1に基づき, VTS $Vsys$より次のアルゴリズムによってそれ以上簡約化できないVTS(これを$Vsys$の既約形(irreducible form)と呼ぶ)を得る.

Algorithm : VTSの簡約化アルゴリズム

input : $Vsys$

output : $Vsys_0$

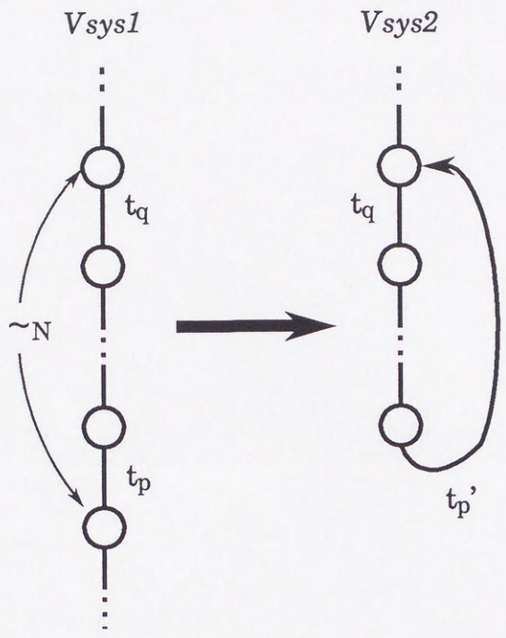


図4-10　簡約化

begin

L: Pを $Vsys$ の有向パスの集合とする；

if reducible path $p \in P$ が存在する

then begin

$p = t_1 \cdots t_k\,(k \geqq 1)$ とする；

$Q := \{n_0\}$;

while $i \leqq k$ **do**

if $to(t_i) \sim_N n$ なる $n \in Q$ が存在

then begin

reduction;

brake

end

else begin

$Q := Q \cup \{to(t_i)\}$;

$i := i + 1$

end ;

reductionで得られた新たなVTSを$Vsys'$と

おく;

$Vsys := Vsys'$;

goto L

end

else $Vsys_0 := Vsys$

end. □

VTS $Vsys_0$が$Vsys$の既約形であり，$Vsys$が文脈から自明の場合，単に$Vsys_0$は既約VTSであるという．

以上のようにして定義されるVTS $Vsys$の既約形$Vsys_0$と$Vsys$との関係について，定理4-1より明らかに，次の系が成り立つ．

【 系4-1 】

VTS $Vsys_0 = <N', G, AS, V, CA, Tr', n_0>$ が $Vsys = <N, G, AS, V, CA, Tr, n_0>$ の既約形であるとき，$Vsys$と$Vsys_0$とは等価である． □

4.4.3 最小TreeLOTOS

この節では，前節で得られた既約VTSから，最小TreeLOTOSを得る方法について述べる．

TreeLOTOSを，LOTOSのサブセットとして次のように定める．

【 定義4-18 】TreeLOTOS

LOTOSのシンタクスを“choice operator”, “action prefix operator”及び“process instantiation”に制限したLOTOSのサブセットをTreeLOTOSという． □

既約VTSにおいて，表4-2のような対応をつけることによってこれを直接的にTreeLOTOSの動作式に変換することができる．この対応関係をLOTOS生成規則と呼ぶ．

【 LOTOS生成規則 】

既約VTS $Vsys_0 = <N', G, AS, V, CA, Tr', n_0>$ よりLOTOS動作式B'を得るための対応規則を表4-2のように定める． □

このLOTOS生成規則に基づいて，最小TreeLOTOSを次のように定める．

【 定義4-19 】最小TreeLOTOS

既約VTS $Vsys_0 = <N', G, AS, V, CA, Tr', n_0>$ よりLOTOS生成規則に基づいて変換して得られる動作式B'と，ASから成るTreeLOTOS $TLS = <B', AS>$を最小TreeLOTOSと定義する． □

最小TreeLOTOSの作り方と命題4-2より，最小TreeLOTOSと既約VTSとの関係について次の命題が成り立つ．

【 命題4-6 】

既約VTS $Vsys_0 = <N', G, AS, V, CA, Tr', n_0>$ より得られる最小TreeLOTOSを$TLS = <B', AS>$とする．このとき，$Sys_1 \sim_S Sys_2$が成り立つ．ここで，

$$Sys_1 = TS_{CLS}(B'),\ Sys_2 = TS_{\iota}(Vsys_0)$$ □

命題4-2, 系4-1, 命題4-6より，仕様と最小TreeLOTOSとの関係について次の定理が成り立つ．

【 定理4-2 】

LOTOS仕様$CLS = <AS, BS>$に関連した閉じた動作式をBとする．この時$VTS_{CLS}(B)$の既約形より得られる最小TreeLOTOSを$TLS = <B', AS>$とおくと

$$B \sim_S B'$$

である． □

既約VTS	TreeLOTOS
状態	Behaviour式
条件 action	条件 action
actionの系列	action prefix expression
Tree の分岐	choice expression
ループ	processs instantiation 及び process 抽象

表4-2　既約VTSとTreeLOTOSの対応

4.5 適用例

以上のことを実際の仕様例に適用した例を示す. 次のような仕様を考える. (データ型については省略)

```
specification EXAMPLE [U,L] : noexit

behaviour

V?m:int[0≦m≦2] ; P0[L,U](m)

where

   process P0[L,U](m:int) : noexit :=

      U?x:mes?v:int ; U!nak ; P0[L,U](m)

   [] U?x:mes?v:int ;

         (  i ; U!nak ; P0[L,U](m)

         [] L!x[v=m] ; U?x:mes?v:int ;

            (  L!x[v=m] ; P0[L,U](m)

            [] i ; U!nak ; P0[L,U](m)

         ) )

   endproc

endspec
```

この仕様例より従来の[ISO 89a]によるLTSの導出体系を適用すると，状態数が166のLTSが得られる．これに対して，4.3で与えた導出体系を適用することによって図4-11(a)のような状態数8のVTS Vsysが得られる．ここで，例えば変数x_2はその全ての出現をx_1で置き換えてもその意味は変わらない．このようにして得られる$Vsys$の変数既約形$Vsys'$は同図(b)の様になる．

ここで，同図(b)の状態*1*と状態*4*に注目する．このとき，$Ind(1)=Ind(4)=\{x,v\}$である．さらに，それぞれの状態を具現化した$RTS_\iota(Vsys')=<S_0,A_0,T_0,s0>$(定義4-8)上の状態集合$B1,B4\subseteq S_0$を

$B1=\{(1,\iota[m\leftarrow j,v\leftarrow k,x\leftarrow l])|(1,\iota[m\leftarrow j,v\leftarrow k,x\leftarrow l])\in S_0,j,k\in int,l\in mes\}$,

$B4=\{(4,\iota[m\leftarrow j,v\leftarrow k,x\leftarrow l])|(4,\iota[m\leftarrow j,v\leftarrow k,x\leftarrow l])\in S_0,j,k\in int,l\in mes\}$

と表すと，$B1$と$B4$の要素間で，

$(1,\iota[m\leftarrow j,v\leftarrow k,x\leftarrow l])\sim_S(4,\iota[m\leftarrow j,v\leftarrow k',x\leftarrow l'])$かつ

$\iota[m\leftarrow j,v\leftarrow k,x\leftarrow l]\sim_{(1,4)}\iota[m\leftarrow j,v\leftarrow k',x\leftarrow l'])$

が成り立つ．例えば，

$(1,\iota[m\leftarrow 0,v\leftarrow 0,x\leftarrow mes1])\sim_S(4,\iota[m\leftarrow 0,v\leftarrow 0,x\leftarrow mes2])$かつ

$\iota[m\leftarrow 0,v\leftarrow 0,x\leftarrow mes1]$と$\iota[m\leftarrow 0,v\leftarrow 0,x\leftarrow mes2]$において$m$の値が等しい(すなわち

$\iota[m\leftarrow 0,v\leftarrow 0,x\leftarrow mes1]\sim_{(1,4)}\iota[m\leftarrow 0,v\leftarrow 0,x\leftarrow mes2])$.

よって，定義4-17に基づき状態*1*と状態*4*について$1\sim_N 4$である．

この関係に基づき，$Vsys'$から簡約化アルゴリズムにより同図(c)のような

状態数5の既約VTS $Vsys_0$が得られる．例えば同図(b)の有向パス

$(0,V?m{:}int[0 \leqq m \leqq 2],1)(1,U?x{:}mes?v{:}int,2)$

$(2,L!x[v=m],4)(4,U?x{:}mes?v{:}int,5)(5,L!x[v=m],1)$

において，簡約化によりこのパスが

$(0,V?m{:}int[0 \leqq m \leqq 2],1)(1,U?x{:}mes?v{:}int,2)$

$(2,L!x[v=m],1)$

となる．

最後に，4.4で与えたLOTOS生成規則の適用によりこの$Vsys_0$から得られる最小TreeLOTOSは次のようになる．

specification *EXAMPLE__Tree* [*U*,*L*] : **noexit**

behaviour

$V?m{:}int[0 \leqq m \leqq 2]$; *P0*[*L*,*U*](*m*)

where

process *P0*[*L*,*U*](*m:int*) : **noexit** :=

$U?x{:}mes?v{:}int$; $U!nak$; *P0*[*L*,*U*](*m*)

[] $U?x{:}mes?v{:}int$;

(**i** ; $U!nak$; *P0*[*L*,*U*](*m*)

[] $L!x[v=m]$; *P0*[*L*,*U*](*m*))

endproc

endspec

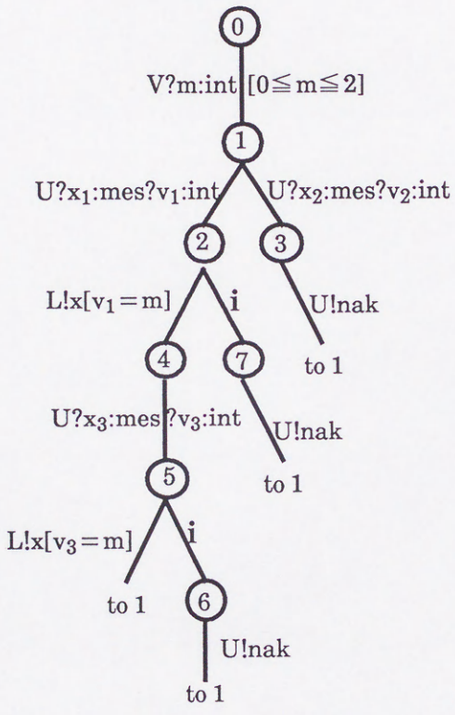


図4-11(a)　仕様例から得られるVTS *Vsys*

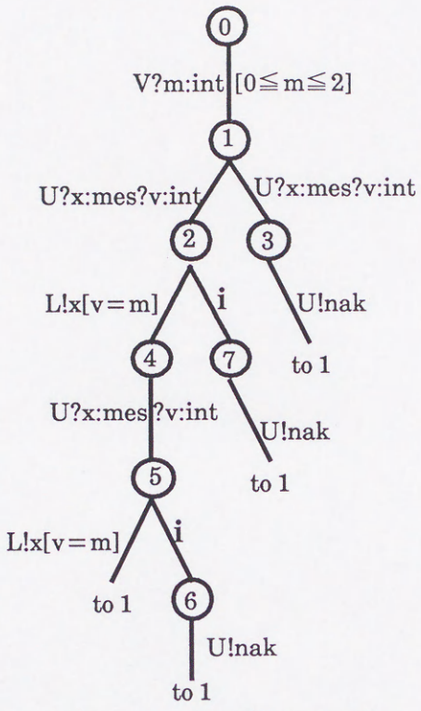


図4-11(b)　Vsys の変数既約形 *Vsys'*

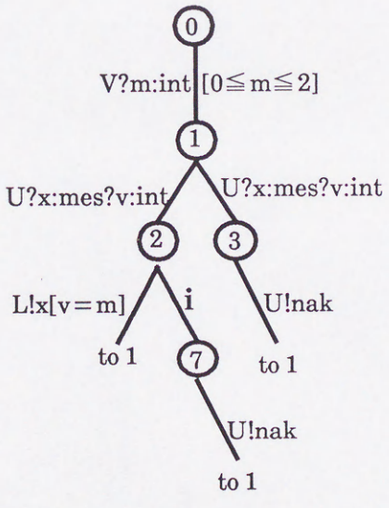


図4-11(c)　既約VTS *Vsys0*

4.6 LOTOS仕様からの試験系列生成支援システムの構築

本節では，前節までの結果に基づいて，LOTOS仕様からの試験系列の生成をユーザに支援するためのシステムの構築を行う．

4.6.1 システムの構成

本支援システムは，図4-12のように

(1) VTS生成機構

(2) 意味解釈機構

(3) 等価性判定機構

(4) 簡約化機構

(5) VTS / LOTOS変換機構

の5つの機能要素から成る．

まず，対象となるLOTOS仕様より，VTS生成機構によってVTSを生成する．次に，このVTSを簡約化機構によって簡約化するために必要な情報を得るために，VTSから意味解釈機構によってLTSを生成する．そして，LTSに関する最小化アルゴリズム[Kami 89]に基づく等価性判定機構により，等価性情報を得る．この等価性情報を用いて，VTS生成機構によって生成されたVTSより簡約化機構によって最小VTSを生成する．最後に，この最小VTSよりVTS / LOTOS変換機構によって最小TreeLOTOSを生成する．本システムの

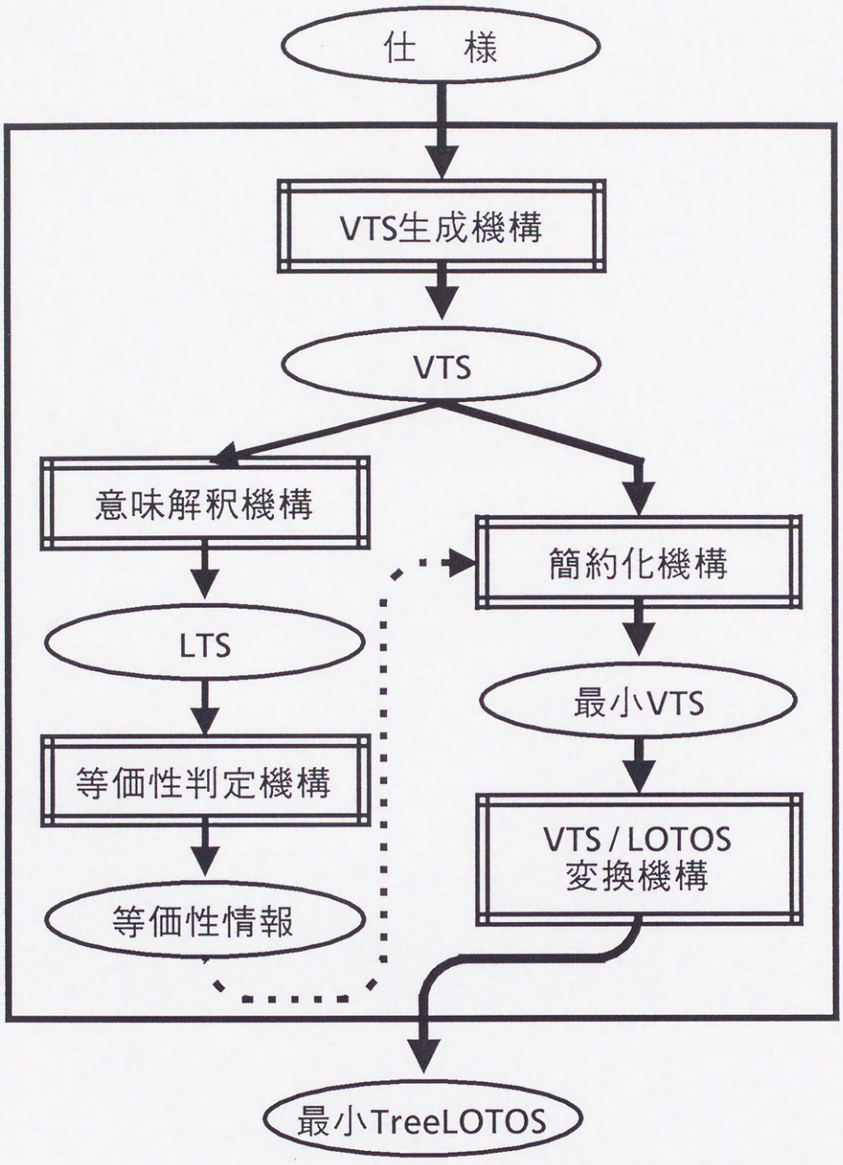


図4-12　システムの構成

出力となる試験スイートは，この最小TreeLOTOSである．

以下では，これらの機能要素に対する基本的な考え方及び設計を与える．

4.6.2　VTS生成機構

与えられたLOTOS仕様のテキストファイルに対して，対応するVTSを生成する．これは，図4-13に示すように，仕様解析機構と仕様展開機構から構成される．

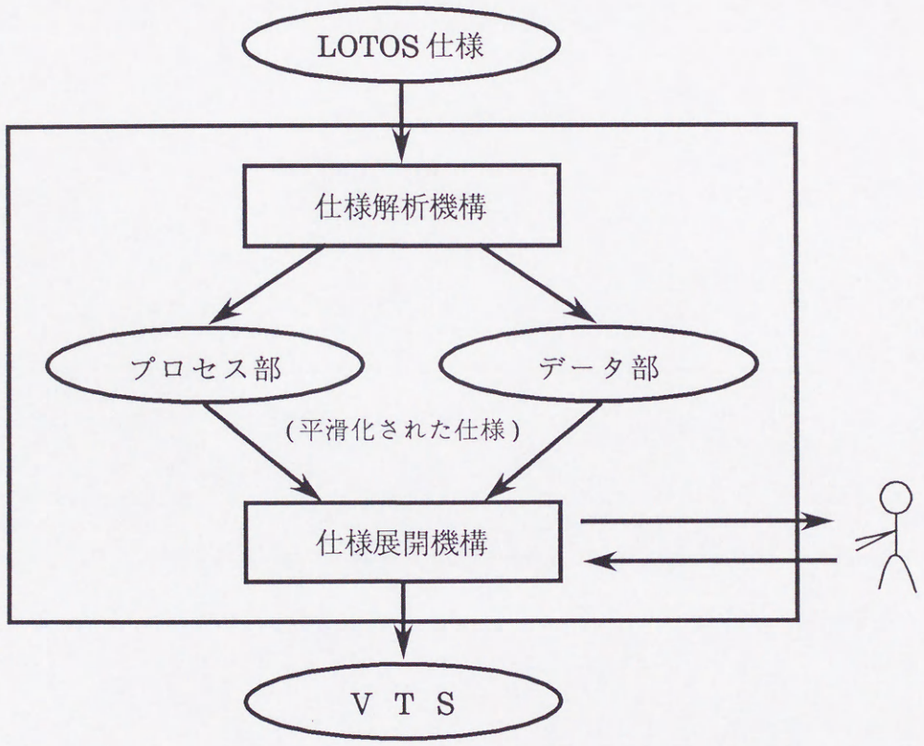


図4-13　VTS生成機構の構成

(1) 仕様解析機構

LOTOS仕様(付録A)を入力とし，字句解析・構文解析・静的意味解析を行い，データ部及びプロセス部から成る平滑化された仕様を出力する．ここで，平滑化された仕様とは，LOTOSの構文規則を満たし，データ部については

(i)　データ型はまとめて一つで表現

(ii)　名前(オペレーション名, 変数名, ソート名)は各クラス毎にグローバルにユニーク

(iii)　等式は前置記法の形で表現

(iv)　変数名にはそのソート情報を付加する

等の性質を満たし, また, プロセス部については

(i)　プロセスはばらばらに定義

(ii)　プロセス名等の名前は各カテゴリ毎にグローバルにユニーク

(iii)　各プロセスの定義動作式は木構造に対応する形式で表現

等の性質を満たすものである. ここでは, 主として以下のことを行う.

(a)　コメントの除去

(b)　構文要素の分離と識別

(c)　名前(識別子)の定義オカレンスと適用オカレンスの検出

(d)　定義オカレンスのスコープの導出

(e)　スコーピング規則を満足するかどうかの判定

(f)　名前のユニーク化

(g)　プロセス部とデータ部それぞれの平滑化表現を生成

(2) 仕様展開機構

平滑化された仕様を入力とし, ユーザとの対話を通して, VTSを出力す

る．すなわち，平滑化された仕様の初期プロセス定義の動作式(仕様のトップレベルの動作式)を入力とし，4.3.2で定めた導出体系の順次的な適用によりVTSを生成する．この時，プロセス部に出現するデータ表現の解釈(項の値の解釈)に当たってはその都度ユーザに支援を求めることにより行う．

4.6.3 意味解釈機構

VTSを入力とし，4.3.3で定めた導出体系の順次的な適用によりLTSを出力する．この時，VTSの遷移に出現するデータ表現の解釈(項の値の解釈)に当たってはその都度ユーザに支援を求めることにより行う．

4.6.4 等価性判定機構

LTSを入力とし，LTSの最小化アルゴリズム[Kami 90]に従って最小化を行う．この時得られるLTSの各状態間の等価性に関する情報を等価性情報として出力する．

4.6.5 簡約化機構

VTSを入力とし，等価性情報を用いてVTSの簡約化を行う．これは，図4-14に示すように，状態等価性判定機構とVTS簡約化機構から構成される．

(1) VTS簡約化機構

VTSを入力とし，4.4.2で示したVTSの簡約化アルゴリズムに従って，最

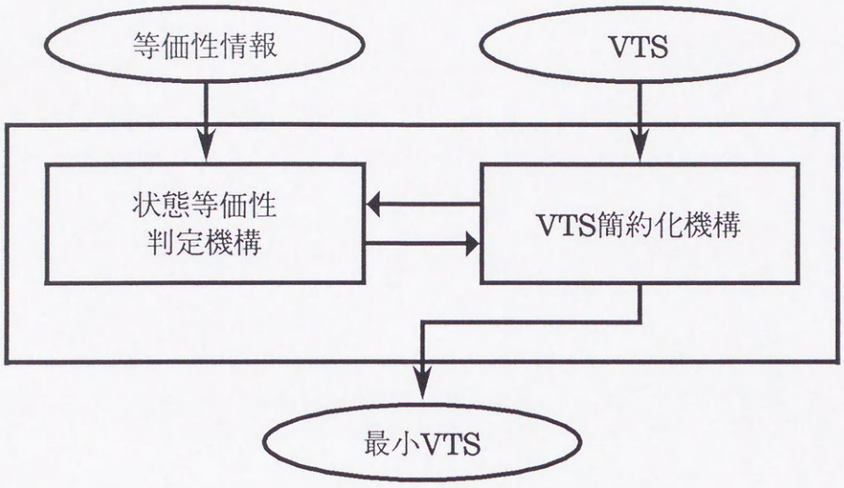


図4-14 簡約化機構の構成

小VTSを生成する．この時，VTSの状態の等価性の判定は状態等価性判定機構を呼び出すことによって行う．

(2) 状態等価性判定機構

VTS簡約化機構に呼び出されることにより起動され，その時渡されるVTSの2つの状態の番号より，等価性情報を用いて4.4.2で定めたVTSの状態間の関係"$\sim_N$"にあるかどうかを調べ，その判定結果をVTS簡約化機構に返す．

4.6.6 VTS / LOTOS変換機構

最小VTSを入力とし，4.4.3で示したLOTOS生成規則に従い最小TreeLOTOSを生成する．

4.6.7　実行例

現在, 以上の設計に従った試験スイートの自動生成支援システムを構築中である. システムは, Sun4 ワークステーションのUNIX上で, C言語を用いてプログラミングを行っている. 以下に, 本システムの実行例を示す.

まず, 入力となるLOTOS仕様を記述した例が図4-15である.

この仕様例を入力として, 4.6.2のVTS生成機構の出力として得られるVTSを示した例が, 図4-16の左側である. さらに, このVTSを入力として, 4.6.5の簡約化機構により得られる最小VTSの例が同図の右側である. 最後に, この最小VTSより4.6.6のVTS/LOTOS変換機構によって得られる最小TreeLOTOSの例が図4-17の右側である. 尚, 同図左側は, 比較のために, 図4-15と同じ仕様例を示したものである.

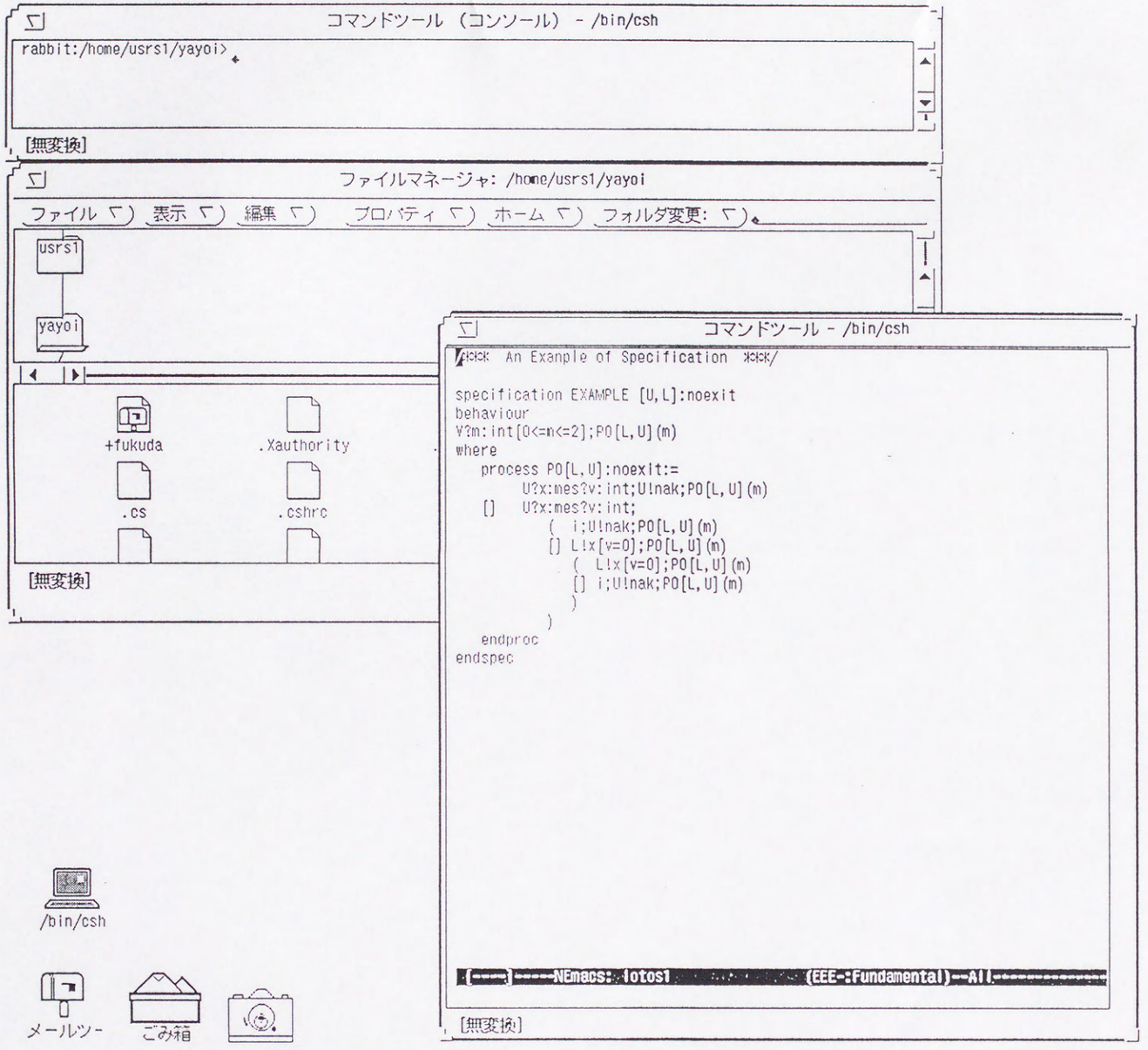


図4-15　LOTOS仕様例

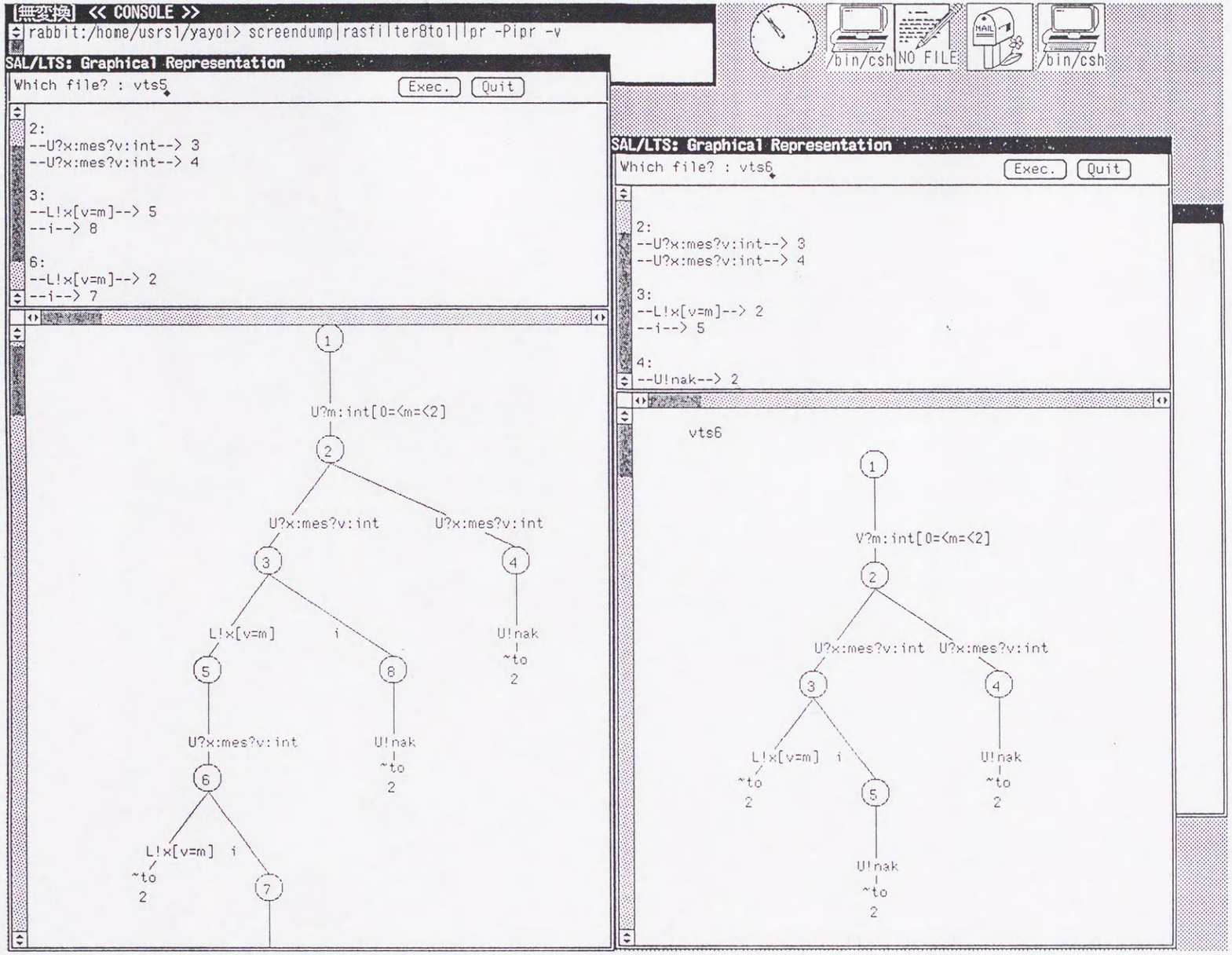


図4-16　VTS及び最小VTS

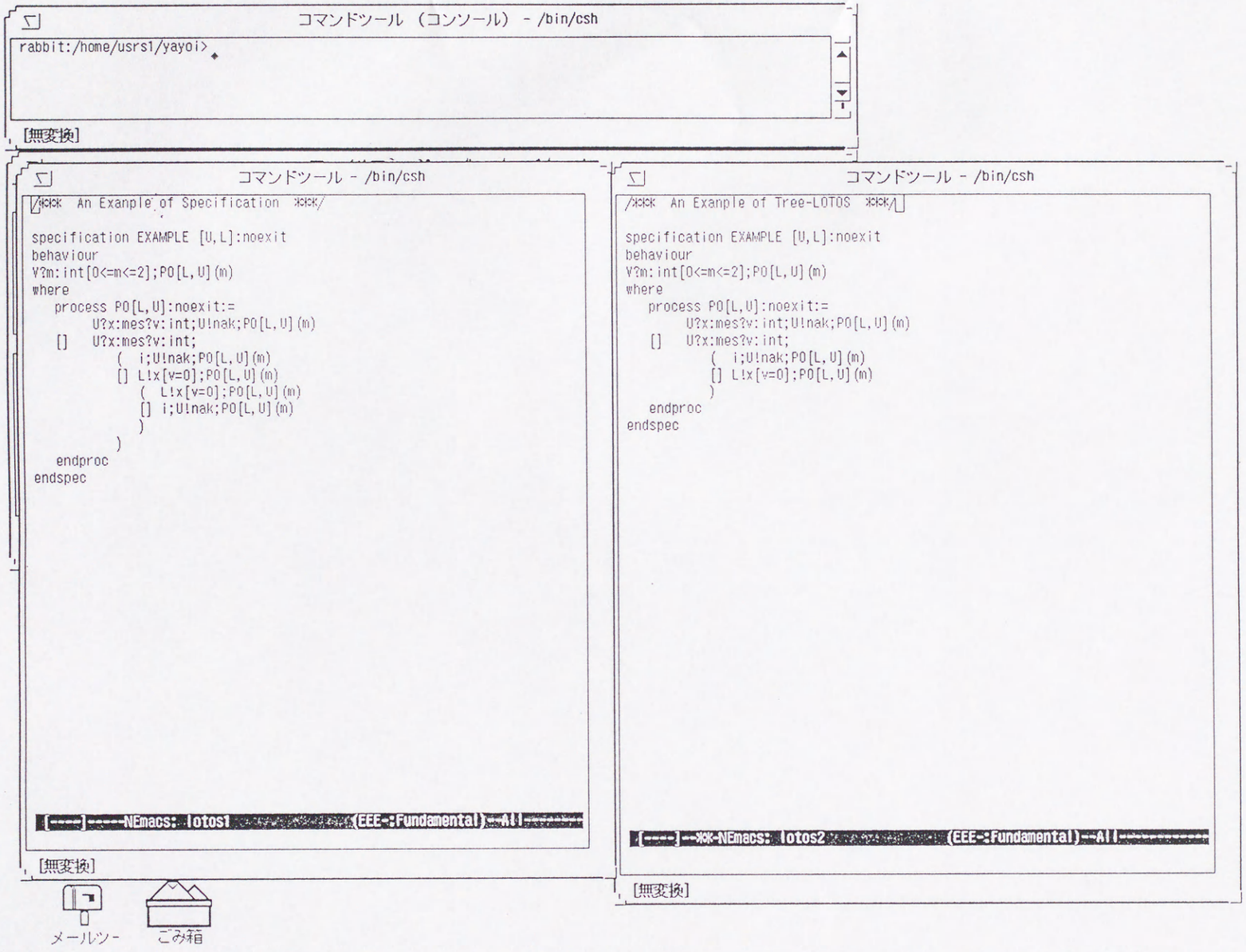


図4-17　LOTOS仕様例及び最小TreeLOTOS

4.7 まとめ

本章では，データ構造を扱ったLOTOS仕様からの試験系列の自動生成の形式的手法を提案した．ここでは，試験系列を試験スイートとして定め，より実用的な試験スイートを得るために，変数付きLTSであるVTSを導入した．そして，VTSの導出体系とVTSの意味を与える具現化規則を定め，この導出体系がLOTOSで定められているLTSの導出体系と矛盾がないことを示した．さらに，より効率的な試験スイートを得るために，弱bisimulation等価の概念によるVTSの簡約化を行った．ここでは，VTSの等価性と既約形を定め，既約形が等価性を保存することを示した．

以下に，本手法の評価として，VTSを導入したことに関して考察する．ここでは次のような場合を考える．すなわち，VTS $Vsys$ を，深さがDで各枝に変数が1つづつ現れるような2分木と仮定する(図4-18)．ここで，

$Vsys$ 上のパスの数(すなわち，VTS上の試験系列数)：$|P_V|$

$Vsys$ を具現化したLTS $TS_t(Vsys)$ 上のパスの数

(すなわち，LTS上の試験系列数)：$|P_L|$

VTS上の試験系列数とLTS上の試験系列数の比：R

とおくと，これらの数は次のように求められる．ただし，

V：$Vsys$ の変数のとり得る値の数の最大値

とし，初期値割当てιは任意に与えられるものとする．

$|P_V|=2^D$

$|P_L|=2^D \cdot V^D$

$R=|P_V|/|P_L|=1/V^D$

例えば，図4-19のようなVTS $Vsys_{ex}$では，D=3, V=8である．従って，この場合の試験系列数の比Rは，$R=1/512 \simeq 0.00195$となる．

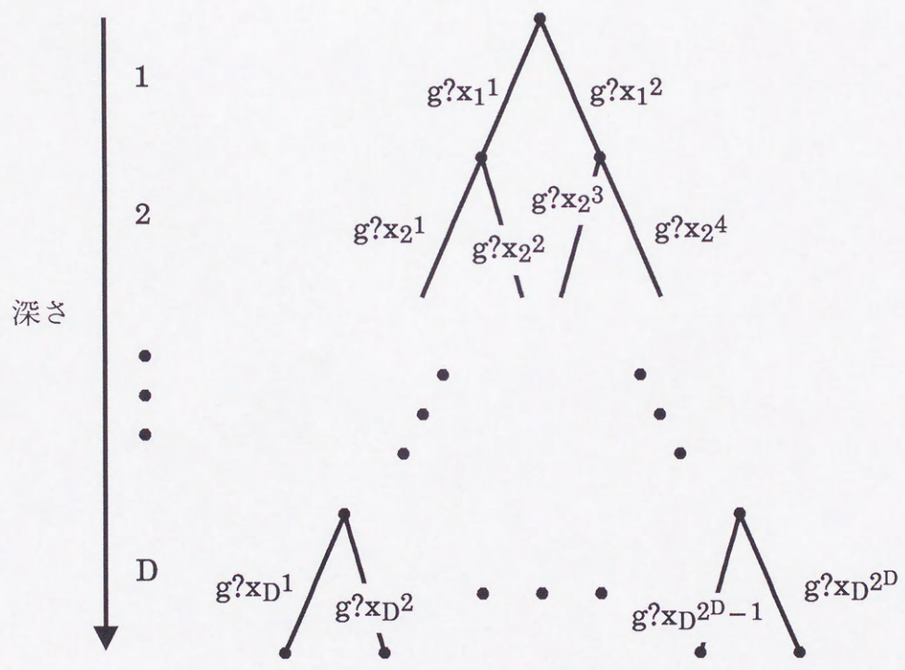


図4-18　深さDの2分木 *Vsys*

尚，4.4においてVTSの変数規約形を仮定したが，これは本手法が一番有効な場合を設定する為であり，本手法の適用のための条件ではない．

本手法では，試験スイートをLOTOSの構文を制限したサブセットであるTreeLOTOSによって表した．このTreeLOTOSは，木状構造であるため，ISO

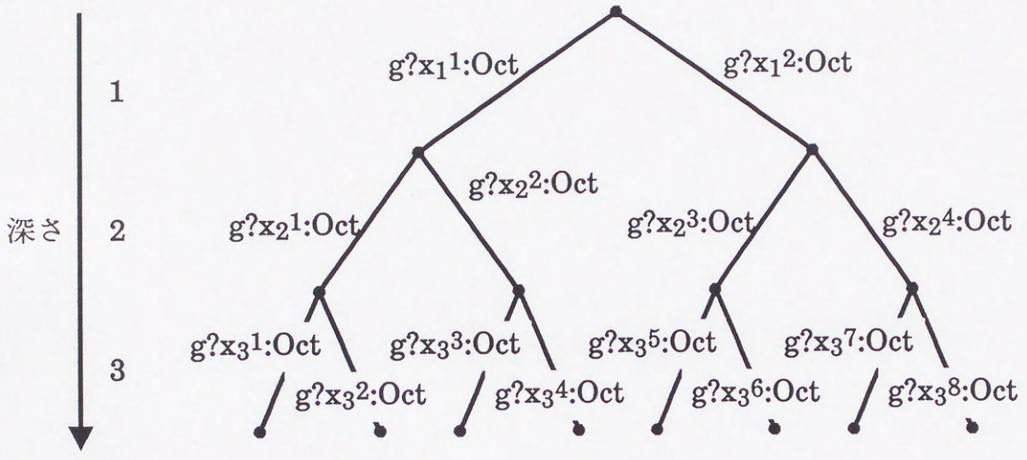


図4-19　$Vsys_{ex}$

で規格化が進められている試験スイート記述言語TTCN[ISO 89c]とその構造がよく一致する．そこで，このTreeLOTOSをTTCN表現に変換して表すことが考えられる．

現在，本手法を用いた試験スイートの自動生成の支援システムを構築中である．

今後の課題としては，効率(長さ，計算量など)を含め，具体的に本方法が有効に適用可能なプロトコルの種別，規模，複雑さなどに関する研究の他，支援システムの構築，大規模なシステムの記述に対する本方法の有効性の確認，本方法で得られる試験スイートを用いた試験実施の支援方法の研究等がある．

第5章　結論

本論文では，通信システムの試験における試験系列の生成に関して

(Ⅰ)　FSMを対象とした相互接続試験系列の生成法

(Ⅱ)　イベントの時間順序に基づくFDTを対象とした形式的な適合性試験系列の生成法

に関する研究を行った．

(Ⅰ)に関する研究としては，第3章において，双方向のチャネルにより相互接続された2つのエンティティの動作を規定するプロトコル仕様を対象とした相互接続試験系列の生成手法を提案した．本手法では，2つのエンティティとその間のチャネルを合成したものを一つのシステムと見なし，そのシステム全体の振る舞いを示すシステム状態グラフを導入した．そして，2つのFSMにより記述されたプロトコル仕様よりシステム状態グラフを生成する手順をルールとして与えた．さらに，ここではチャネルにエラーがある場合を想定したプロトコル仕様に対しても適用できるようこの生成ルールを拡張した．このルールによって生成されたシステム状態グラフ上には，システム全体の各アクション間の前後関係が明確に表されており，このことにより相互接続試験における同期の問題が解決されることを示した．また，この生成ルールを拡張し，このことにより，チャネルにエラーがある場合を想定したプロトコル仕様に対しても適用できることを示した．さらに，拡張したルールにより得られた拡張システム状態グラフを変形し，従来の適合性

試験に対する試験系列生成法が適用可能な形に導き，相互接続試験系列を生成する方法を与えた．本手法を用いることによって，相互接続試験における試験系列生成のコストの削減，試験の信頼性の向上などが期待される．

また，(Ⅱ)に関する研究としては，第4章において，イベントの時間順序に基づくFDTとしてLOTOSを対象とし，弱bisimulation等価の概念を用いて木状の試験系列を生成する形式的な手法を提案した．本手法では，LOTOSの意味表現であるラベル付き遷移システム(LTS)に変数表現を加えた変数付きLTS(VTS)を導入した．このことにより，試験系列に変数表現を取り入れることができ，これによって，表現がコンパクトになるだけでなく，試験実施時における系列の選択に有効な情報を試験系列に与えることができることを示した．さらに，弱bisimulation関係によるLTSの等価性の概念を拡張し，VTS上での状態の等価性を定めた．そして，この等価性に基づくVTSの簡約化の方法を与え，この簡約化を繰り返すことによりVTSを既約形に変換した．これによって，より効率的な試験系列を生成することができることを示した．また，本手法に基づく試験系列生成支援システムの構築を行って，その有効性を示した．

謝　辞

本研究は，東北大学応用情報学研究センター 野口正一 教授 の御指導のもとに行うことができました．同教授には，研究面での熱心な御指導のみならず，精神面での多大な御支援を頂きました．ここに深く感謝致します．

また，研究内容についてその技術的な進め方からまとめ方にいたるまで熱心な御指導を頂きました 東北大学工学部 白鳥則郎 教授に深く感謝致します．

さらに，東北大学電気通信研究所 高橋薫 技官には，日頃からゼミ等において，筆者の直接的な細部の研究内容について熱心な御討論と有益な助言を頂きました．本博士論文における研究成果は，まさに同氏との討論なくしては得られないものでありました．深く感謝致します．

本博士論文の審査をして頂きました 東北大学工学部 高木相 教授に深く感謝致します．

試験系列生成支援システムの作成に従事して頂きました ㈱高度通信システム研究所 大友弥生 さんに感謝致します．

研究生活を送るにあたりいろいろとお世話になりました 東北大学電気通信研究所 野口研究室 及び 東北大学工学部情報工学科 白鳥研究室，㈱高度通信

システム研究所の皆様に感謝致します.

最後に,研究内容の良き相談相手であるとともに筆者の生活面,精神面の全てにおいて支えとなってくれました 妻 朴美娘に,そして生活面,精神面でこれまで支えてくれました両親に,心から感謝致します.

参考文献

[Boch 78] G.V.Bochmann : "Finite state description of communication protocols", Computer Networks, 2, pp.361-371, (1978).

[Boch 80a] G.V.Bochmann and C.A.Sunshine : "Formal Methods in Communication Protocol Design", IEEE Trans. Commun., COM-28, 4, pp.624-631, (1980).

[Boch 80b] G.V.Bochmann : "A General Transition Model for Protocols and Communication Services", IEEE Trans. Commun., COM-28, 4, pp.643-650, (1980).

[Boch 82] G.V.Bochmann, E.Cerny, M.Gagné, C.Jard, A.Léveillé, C.Lacaille, M.Maksud, K.S.Raghunathan and S.Sarikaya : "Experience with Formal Specifications Using an Extended State Transition Model", IEEE Trans. Commun., COM-30, 12, pp.2506-2513, (1982).

[Boch 87] G.V.Bochmann : "Usage of Protocol Development Tools : The Results of a Survey", *in* Protocol Specification, Testing and Verification Ⅶ, North-Holland, pp.139-161, (1987).

[Bolo 87a] T.Bolognesi and E.Brinksma : "Introduction to the ISO Specification Language LOTOS", Computer Networks and ISDN Systems, 14, North-Holland, (1987).

[Bolo 87b] T.Bolognesi and S.A.Smolka : "Fundamental Results for the Verification of Observational Equivalence : a Survey", *in* Protocol Specification, Testing and Verification Ⅶ, North-Holland, pp.161-178, (1987).

[Bran 83] D.Brand and P.Zafiropulo : "On communicating finite state machines", J.ACM, 30, pp.323-342, (1983).

[Bria 86] J.P.Briand, M.C.Fehri, L.Logrippo and A.Obaid : "Executing LOTOS Speccifications", *in* Protocol Specification, Testing and Verification Ⅵ, North-Holland, pp.73-84, (1986).

[Brin 84] E. Brinksma and G.Karjoth : "A Specification of the OSI transport service in LOTOS", *in* Protocol Speecification, Testing and Verification Ⅳ, North-Holland, pp.227-251, (1984).

[Brin 85] E. Brinksma : "A Tutorial on LOTOS", *in* Protocol Speecification, Testing and Verification Ⅴ, North-Holland, pp.171-194, (1985).

[Brin 86] E.Brinksma, G.Scollo and C.Steenbergen : "LOTOS Specification, Their Implementation and Their Tests", *in* Protocol Specification, Testing and Verification Ⅵ, North-Holland, pp.9-13, (1986).

[Broo 84] S.D.Brookes, C.A.R.Hoare and A.W.Roscoe : "A theory of communicating sequential processes", J.ACM, 31, 3, pp.560-599, (1984).

[Carc 86] V.Carchiolo, A.Faro, O.Mirabella, G.Pappalardo and G.Scollo : "A LOTOS Specification of the PROWAY Highway Service", IEEE Trans. Comput. vol.C-35, no.11, pp.949-968, (1986).

[CCIT 84] CCITT : "Functional Specification and Description Language (SDL)", CCITT Recommendations Z. 100 - Z. 104, (1984).

[Chow 78] T. Chow : "Testing software design modeled by finite-state machines," *IEEE Trans. Software Eng.*, vol. SE-4, pp. 178-187, (1978).

[Dant 80] A.S.Danthine : "Protocol Representation with Finite-State Models", IEEE Trans. Commun., COM-28, 4, pp.632-642, (1980).

[Ehri 83] H.Ehrig, et al. : "ACT ONE, An Algebraic Specification Language with Two Levels of Semantics", Bericht - Nr. 83-03, Berlin University, (1983).

[Freu 87] J.Freudenmann : "Development of communication software by stepwise refinement", *in* Protocol Specification, Testing and Verification Ⅶ, North-Holland, pp.391-404, (1987).

[Gilb 87] D.Gilbert : "Executable LOTOS : Using PARLOG to implement an FDT", *in* Protocol Specification, Testing and Verification Ⅶ, North-Holland, pp.281-294, (1987).

[Golt 84] U.Goltz and A.Mycroft : "On the relationships of CCS and Petri nets", Lecture Notes in Computer Science, 172, Springer-Verlag, (1984).

[Gone 70] G. Gonenc : "A method for the design of fault detection experiment," *IEEE Trans. Comput.*, vol. C-19, pp. 551-558, (1970).

[Gotz 86] R.Gotzhein : "Specifying Abstract Data Types with LOTOS", *in* Protocol Specification, Testing and Verification Ⅵ, North-Holland, pp.61-76, (1986).

[Goud 84] M.G.Gouda and Y.T.Yu : "Protocol Validation by Maximal Progress State Exploration", IEEE Trans. Commun., COM-32, 1, pp.94-97, (1984).

[Guer 89] D. Gueraichi and L. Logrippo : "Derivation of test cases for LAP-B from a LOTOS specifications", *in proc. of FORTE '89 (FORMAL DESCRIPTION TECHNIQUES for Distributed Systems and Communications Protocols)* (1989).

[Gutt 77] J.V.Guttag : "Abatract Data Types and the Development of Data Structures", C.ACM, 20, pp.396-404, (1977).

[Gutt 78] J.V.Guttag, E.Horowitz and D.R.Musser : "Abstract Data Types and Software Validation", C.ACM, 21, pp.1048-1063, (1978).

[Hail 85] B.T.Hailpern : "A Simple Protocol Whose Proof Isn't it", IEEEE Trans. Commun., COM-33, no.4, pp.330-337, (1985).

[Hoar 78] C.A.R.Hoare : "Communicating Sequential Processes", C.ACM, 21, 8, pp.666-677, (1978).

[Hogr 90] D. Hogrefe : "Conformance testing based on formal methods," *Proceeding of FORTE '90*, pp. 213-245 (1990).

[Hopc 79] J.Hopcroft and J.D.Ullman : "An introduction to automata theory, languages and computation", Addison-Wesley, (1979).

[Inag 84] 稲垣, 坂部 : "抽象データタイプの代数的仕様記述法の基礎 (1) - (4)", 情報処理, 25, 1 - 9, (1984).

[ISO 89a] ISO : "Information processing systems - open systems interconnection, LOTOS - a formal description technique based on the temporal ordering of observational behaviour", ISO 8807, (1989).

[ISO 89b] ISO : "Information processing systems - open systems interconnection, Estelle - a formal description technique based on an extended state transition model", ISO 9074, (1989).

[ISO 89c] ISO : "OSI conformance testing methodology and framework", ISO 9646 (1989).

[Kami 89] 神長, 高橋, 白鳥, 野口 : "LOTOS 仕様の等価性とその判定法", 信学論 (D-Ⅰ), J72-D-Ⅰ, 5, pp.367-376, (1989).

[Kami 90] 神長, 高橋, 白鳥, 野口 : "LOTOS仕様の効率的な等価性判定法", 信学論(D-1), J-73-D-1, 2, pp. 214-224 (1990).

[Kawa 85] 河岡, 荒木 "プロトコルの形式的記述法", 電子通信学会誌, 68, 2, pp187-192, (1985).

[Lam 84] S.L.Lam and A.U.Shankar : "Protocol verification via projections", IEEE Trans. Software Eng., SE-10, 4, pp.325-342, (1984).

[Ledu 87] G.J.Leduc : "The Intertwining of Data Types and Processes in LOTOS", *in* Protocol Specification, Testing and Verification Ⅶ, North-Holland, pp.123-136, (1987).

[MaQu 77] J.M.MaQuillan and D.C.Walden : "The ARPA network design decisions", Computer Networks, vol.1, pp.243-289, (1977).

[Miln 80] R.Milner : "A Calculus of Communicating Systems", Lecture Notes in Computer Science, 92, Springer-Verlag, (1980).

[Miln 83] R.Milner : "Caluli for Synchrony and Asynchrony", Theoretical Computer Science, 25, North-Holland, pp.267-310, (1983).

[Miln 89] R.Milner : "Communication and concurrency", Prentice Hall (1989).

[Moto 85] 元岡, 苗村, 他 : "大特集 : ネットワークアーキテクチャ (開放型システム相互間接続) の標準化動向", 情報処理, 26, 4, (1985).

[Muss 80] D.R.Musser : "Abstract Data Type Specification in the AFFIRM System", IEEE Trans. Software Eng., SE-6, 1, pp.24-32, (1980).

[Nait 81] S. Naito and M. Tsunoyama : "Fault detection for sequential machines by transition tours," in *Proc. IEEE Fault Tolerant Comput. Conf.*, (1981).

[Najm 87] E.Najm : "A verification orientd specification in LOTOS of the transport protocol", *in* Protocol Specification, Testing and Verification Ⅶ, North-Holland, pp.181-203, (1987).

[Naka 90] 中原彰子, 相田仁, 斎藤忠夫 : "通信プロトコルのテスト系列生成手法に関する一評価", 信学論, vol. J73-B-Ⅰ, No. 11, pp. 853-863 (1990).

[Papp 87] G.Pappalardo : "Experiences with a verification and simulation tool for behavioural languages", *in* Protocol

Specification, Testing and Verification Ⅶ, North-Holland, pp.251-264, (1987).

[Plet 86] U.Pletat : "Algebraic Specifications of Abstract Data Types and CCS : An Operational Junction", *in* Protocol Specification, Testing and Verification Ⅵ, North-Holland, pp.361-372, (1986).

[Quem 87] J.Quemada and A.Fernandez : "Introduction of Quantitative Relative Time into LOTOS", *in* Protocol Speecification, Testing and Verification Ⅶ, North-Holland, pp.105-121, (1987).

[Rayn 87] D.Rayner : "OSI Conformance Testing", Computer Networks and ISDN Systems, 14, North-Holland, (1987).

[Rock 82] A.Rockström and R.Saracco : "SDL - CCITT Specification and Description Language", IEEE Trans. Commun., COM-30, 6, pp.1310-1317, (1982).

[Sabn 88] K. Sabnami and A. Dahbura : "A protocol test generation procedure", *Computer Networks ISDN System*, 15, pp. 285-297 (1988).

[Sait 88] 斎藤, 苗村 : "OSIの実現とその課題 (I) OSI標準化の動向", 情報処理, 29, 9, pp.1023-1031, (1988).

[Sari 84] B.Sarikaya and G.V.Bochmann : "Synchronization and specification issues in protocol testing", IEEE Trans. Commun., COM-32, 4, pp.389-395, (1984).

[Sari 87] B.Sarikaya, G.V.Bochmann and E.Cerny : "A Test Design Methodology for Protocol Testing", IEEE Trans. Software Eng., vol.SE-13, no.5, pp.518-531, (1987).

[Sari 89] B.Sarikaya : "Conformance Testing : Archirtectures and Test Sequences", North-Holland Computer Networks and ISDN Systems, vol.17, pp.111-126, (1989).

[Sato 91] 佐藤文明, 宗森純, 井出口哲夫, 水野忠則 : "有限オートマトンに基づくシステムの試験系列自動生成手法の提案", 信学論, vol. J72-B-Ⅰ, No. 3, pp. 183-192 (1991).

[Scol 86] G.Scollo and M.V.Sinderen : "On the Architectural Design of the Formal Specification of the Session Standards in LOTOS", *in* Protocol Specification, Testing and Verification Ⅵ, North-Holland, pp.3-14, (1986).

[Schw 82] R.Schwartz and P.M.Melliar-Smith : "From State Machines to Temporal Logic : Specification Methods for Protocol Standards", IEEE Trans. Commun., COM-30, 12, pp.2486-2496, (1982).

[Shir 85a] 白鳥, 郷原, 野口 : "EXPA : パータベーション解析に基づく通信プロトコルの検証法", 情処学論, 25, 3, pp.446-453, (1985).

[Shir 85b] 白鳥, 高橋, 野口 : "NESDEL : プロトコル向き仕様記述言語とその応用", 情処学論, 26, 6, pp.1136-1144, (1985).

[Shir 87] 白鳥 : "試験検証技術", 情報処理, 28, 4, pp.403-410, (1987).

[Sidh 89] D. P. Sidhu and T. Leung : "Formal method for protocol testing : a detailed study," *IEEE Trans. Software Eng.*, SE-15, 4, pp. 413-426 (1989).

[Sten 76] N.V.Stenning : "A Data Transfer Protocol", Computer Networks, 1, pp.99-110, (1976).

[Suns 82] C.A.Sunshine, D.H.Thompson, R.W.Erickson, S.L.Gerhart and D.Schwabe : "Specification and Verification of Communication Protocols in AFFIRM Using State Transition Models", IEEE Trans. Software Eng., SE-8, 5, pp.460-489, (1982).

[Tret 89] Tretmans : "Test Case Derivation from LOTOS Specifications", *in proc. of FORTE '89 (FORMAL DESCRIPTION TECHNIQUES for Distributed Systems and Communications Protocols)* (1989).

[Turn 87] K.J.Turner : "An Architectural Semantics for LOTOS", *in* Protocol Specification, Testing and Verification Ⅶ, North-Holland, pp.15-28, (1987).

[Taka 90] 高橋, 神長, 白鳥 : "LOTOS言語の特質と処理系の現状と動向", 情報処理, 31, 1, pp. 35-46 (1990).

[Tana 87] 田中, 他 : "大特集 : 分散処理技術", 情報処理, 28, 4, (1987).

[Toma 83] 当麻喜弘,内藤祥雄,南谷崇 : "順序機械", 岩波書店(昭58).

[Uyar 86] M. U. Uyar and A. T. Duhbura : "Optimal Test Sequence Generation for Protocols : The Chinese Postman Algorithm Applied to Q.931", *IEEE Global Telecommunications Conference* (1986).

[Venk 85] R.C.Venkatraman and T.F.Piatkowski : "A Formal Comparison of Formal Protocol Specification Techniques", in Protocol Specification, Testing, and Verification Ⅴ, North-Holland, pp.401-420, (1985).

[Viss 83] C.A.Visser, R.L.Tenney and G.V.Bochmann : "Formal Description Techniques", *in* Proceedings of the IEEE, 71, 12, pp.1356-1362, (1983).

[West 78] C. H. West : "General technique for communications protocol validation," *IBM J. Res. & Devel.*, vol. 22-4, pp. 394-404 (1978).

[Wirt 71] N.Wirth : "Program development by stepwise refinement", C.ACM, 14, 4, pp.221-227, (1971).

[Yasu 87] 安田, 馬渡, 他 : “情報ネットワーク特集”, 信学誌, 70, 11, (1987).

[Zafi 80] P.Zafiropulo, C.H.West, H.Rudin, D.D.Cowan and D.Brand : “Towards Analyzing and Synthesizing Protocols”, IEEE Trans. Commun., COM-28, 4, pp.651-661, (1980).

[Zimm 80] H.Zimmerman : “OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection”, IEEE Trans. Commun., COM-28, 4, pp.425-432, (1980).

発表論文リスト

(1) 岡崎直宣, 相沢茂喜, 高橋薫, 白鳥則郎, 野口正一 : “ユーザフレンドリーなプロトコル検証システムの構成”, 情報処理学会研究報告(マルチメディアと分散処理), MDP 33-4 (1987).

(2) 岡崎直宣, 相沢茂喜, 高橋薫, 白鳥則郎, 野口正一 : “ユーザフレンドリーなプロトコル検証システム”, 電子情報通信学会交換·情報ネットワーク研究ワークショップ (1987).

(3) 山本博章, 相沢茂喜, 岡崎直宣, 高橋薫, 白鳥則郎, 野口正一 : “ユーザフレンドリープロトコル検証システム”, 電子情報通信学会論文誌(D), vol.J72-D, No.11, pp.2219-2227 (1987).

(4) 岡崎直宣, 高橋薫, 白鳥則郎, 野口正一 : “LOTOS仕様からのテストシーケンスの自動生成”, 情報処理学会第36回(昭和63年前期)全国大会, pp.583-584 (1988).

(5) 岡崎直宣, 高橋薫, 白鳥則郎, 野口正一 : “LOTOS仕様からのテストシーケンスの自動生成”, 電子情報通信学会技術研究報告(交換システム), SSE 88-81, pp.13-18 (1988).

(6) 岡崎直宣, 高橋薫, 白鳥則郎, 野口正一 : “LOTOS仕様からのTTCN表現によるテストシーケンスの自動生成”, 情報処理学会第39回(平成1年後期)全国大会, pp.1973-1974 (1989).

(7) 岡崎直宣, 高橋薫, 白鳥則郎, 野口正一 : “LOTOS仕様からのTTCN表現によるテストシーケンスの自動生成”, 電子情報通信学会技術研究報告(情報ネットワーク), IN 89-25, pp.25-30 (1989).

(8) 岡崎直宣, 高橋薫, 白鳥則郎, 野口正一 : “LOTOS仕様からの効率的な試験系列の自動生成”, 情報処理学会第40回(平成2年後期)全国大会, pp.1431 (1990).

(9) 岡崎直宣, 高橋薫, 白鳥則郎, 野口正一 : “LOTOS仕様からの効率的な試験系列の自動生成”, 情報処理学会研究報告(マルチメディアと分散処理), MDP 46-7, pp.49-56 (1990).

(10) 岡崎直宣, 高橋薫, 白鳥則郎, 野口正一 : “LOTOS仕様からの効率的な試験系列の生成法”, 電子情報通信学会論文誌(B-Ⅰ), vol.J74-B-Ⅰ, No.10, pp.733-747 (1991).

(11) 大友弥生, 岡崎直宣, 太田正孝 : "LOTOS仕様からの試験系列生成支援システムの構築", 情報処理学会第43回(平成3年後期)全国大会, No.5, pp.269-260 (1991).

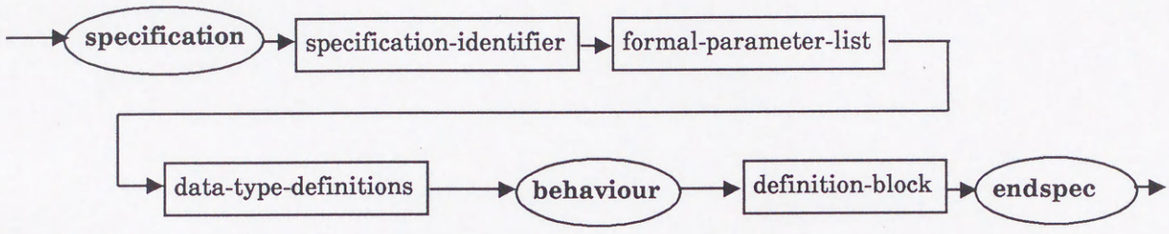
(12) 岡崎直宣, 高橋薫, 白鳥則郎, 野口正一 : "通信システムにおける相互接続試験系列生成法", 電子情報通信学会技術研究報告(情報ネットワーク), IN 91-107, pp.39-44 (1991).

(13) 岡崎直宣, 高橋薫, 白鳥則郎, 野口正一 : "通信システムにおける相互接続試験系列生成法", 電子情報通信学会論文誌(B-Ⅰ), 投稿中.

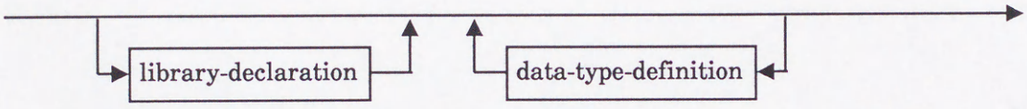
付録　LOTOS構文規則

本手法で扱うLOTOS仕様の構文を示す．これは，基本的には省略記法的な構文を取り除いた，本来のLOTOS仕様の構文[ISO 89a]のサブセットである．

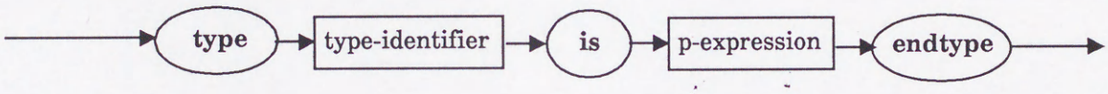
specification



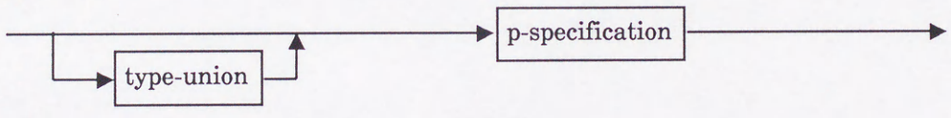
data-type-definitions



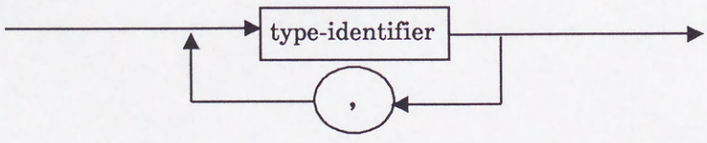
data-type-definition



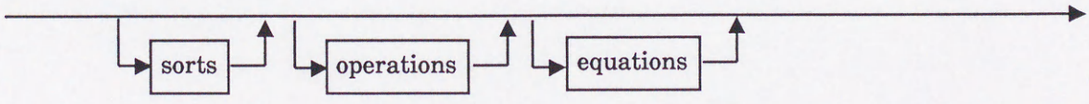
p-expression



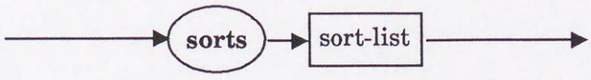
type-union



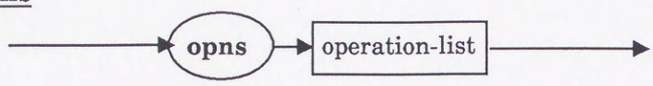
p-specification



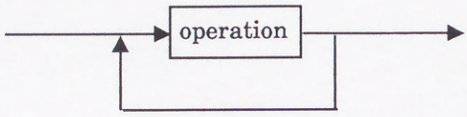
sorts



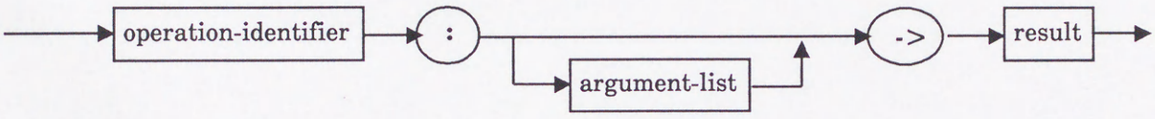
operations



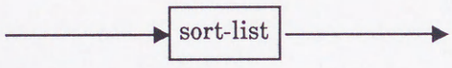
operation-list



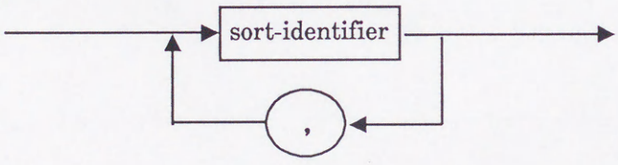
operation



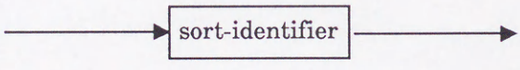
argument-list



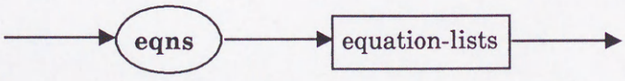
sort-list



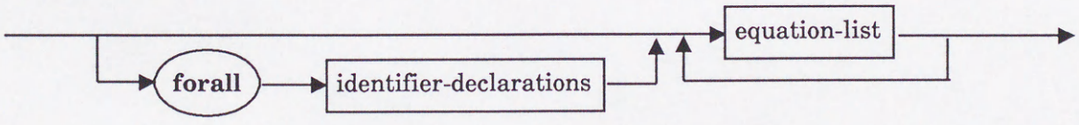
result



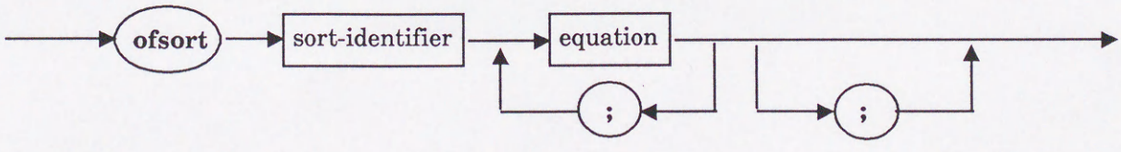
equations



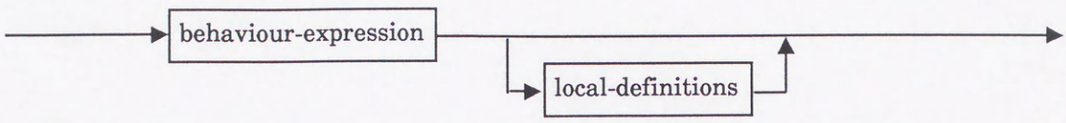
equation-lists



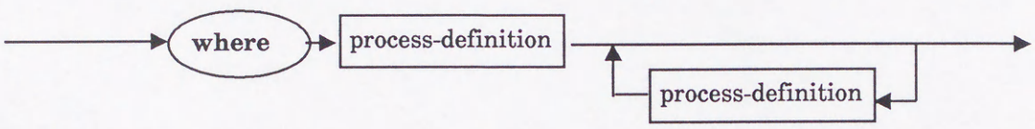
equation-list



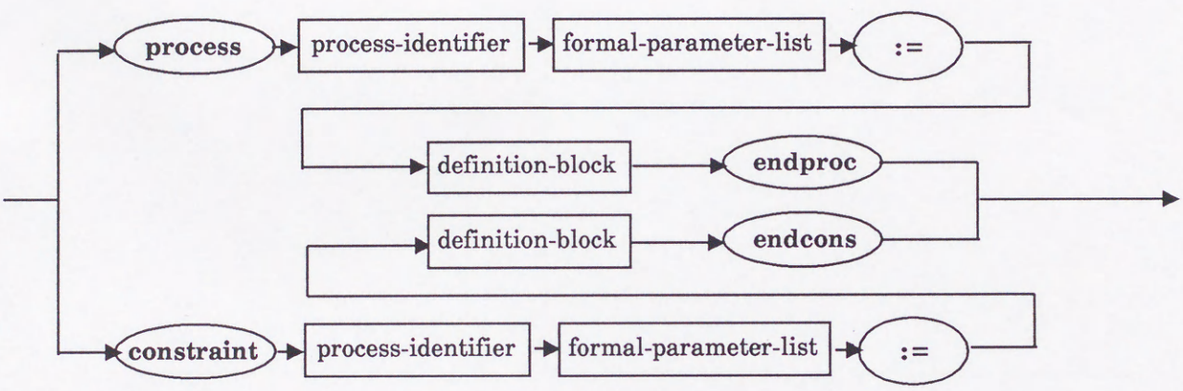
definition-block



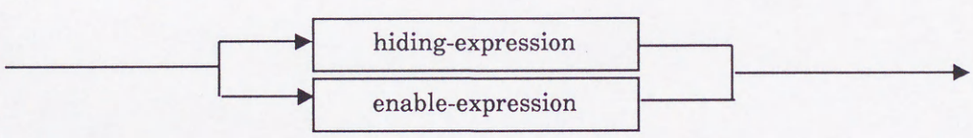
local-definitions



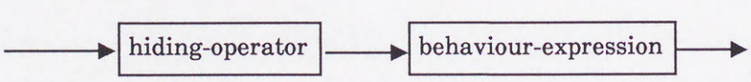
process-definition



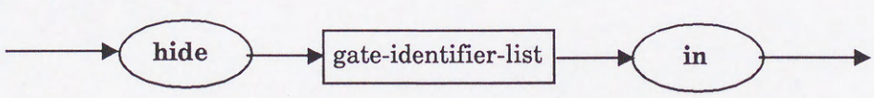
behaviour-expression



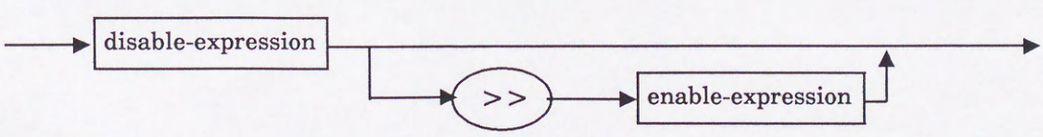
hiding-expression



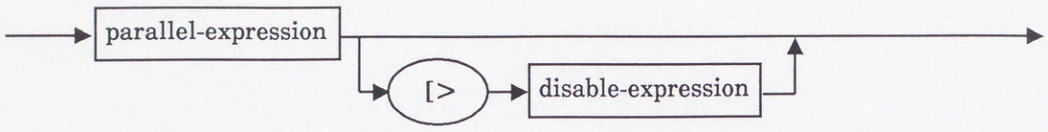
hiding-operator



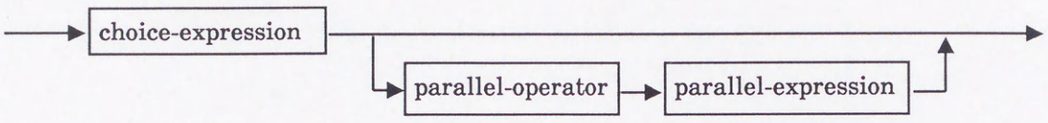
enable-expression



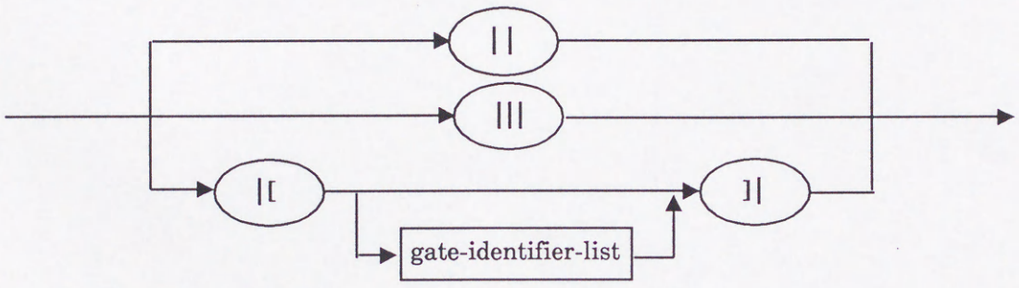
disable-expression



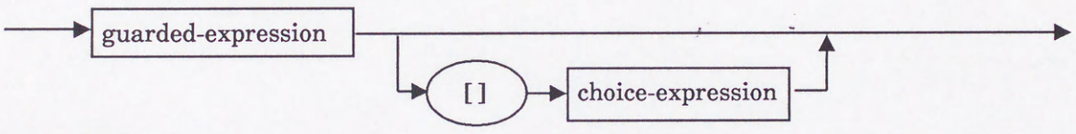
parallel-expression



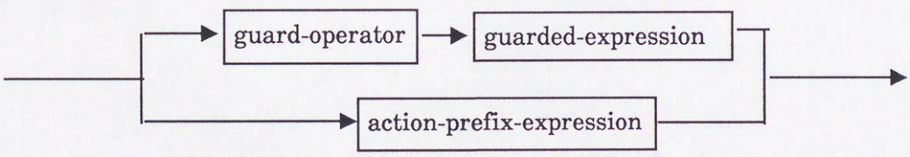
parallel-operator



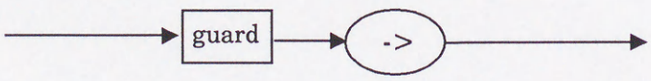
choice-expression



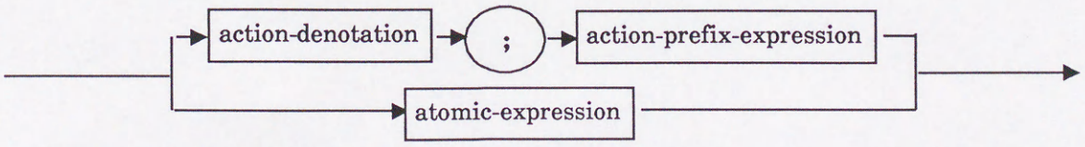
guarded-expression



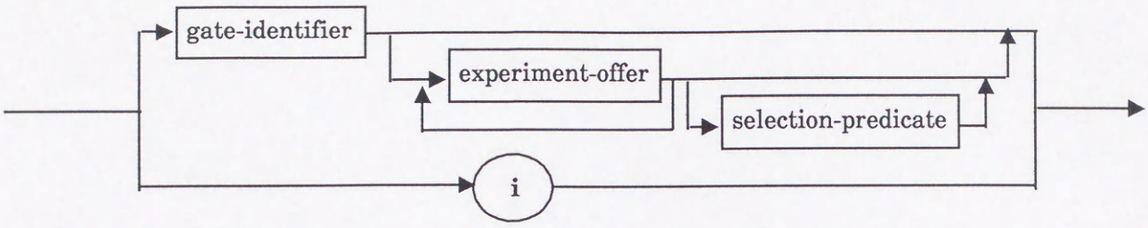
guard-operator



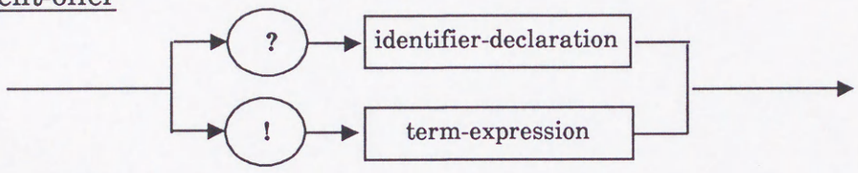
action-prefix-expression



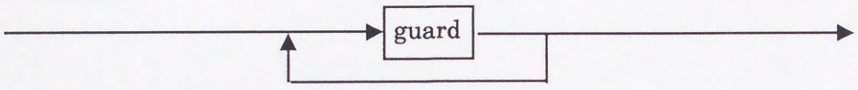
action-denotation



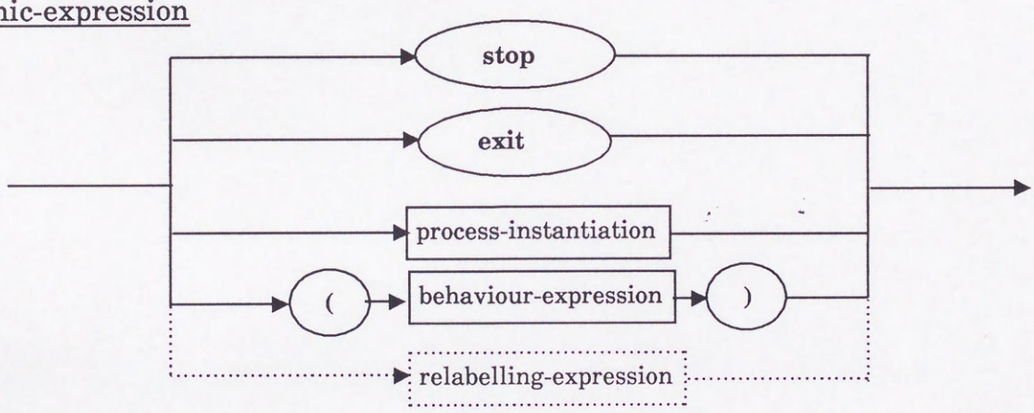
experiment-offer



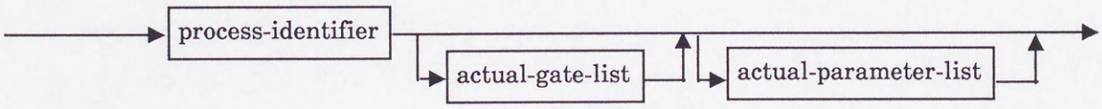
selection-predicate



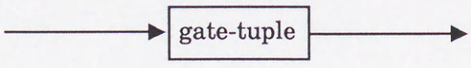
atomic-expression



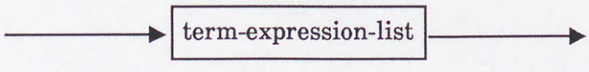
process-instantiation



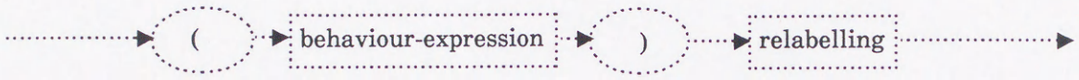
actual-gate-list



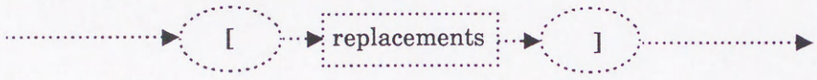
actual-parameter-list



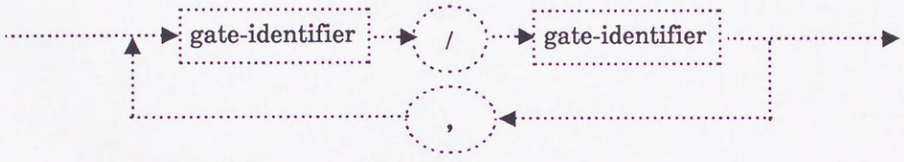
relabelling-expression



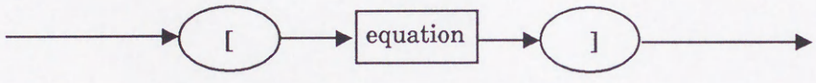
relabelling



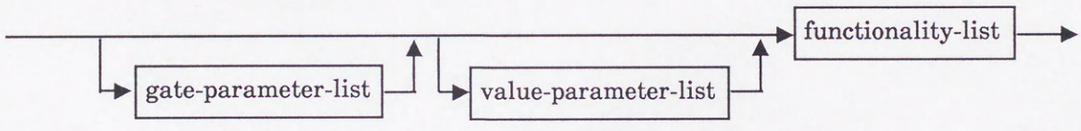
replacements



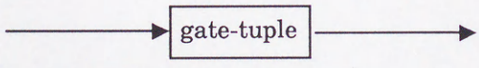
guard



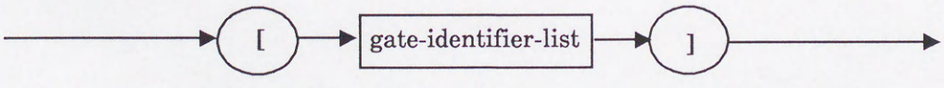
formal-parameter-list



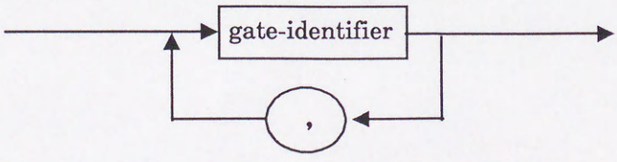
gate-parameter-list



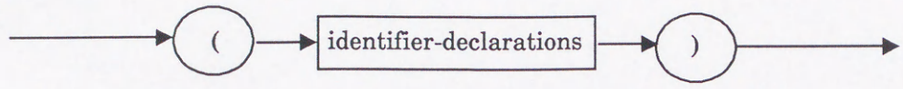
gate-tuple



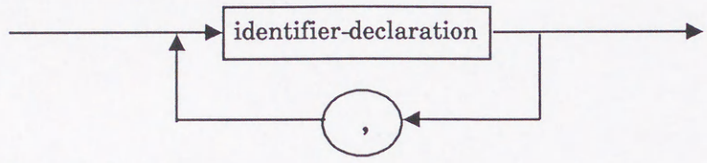
gate-identifier-list



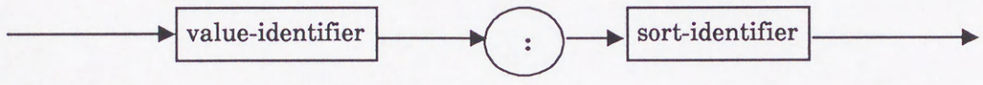
value-parameter-list



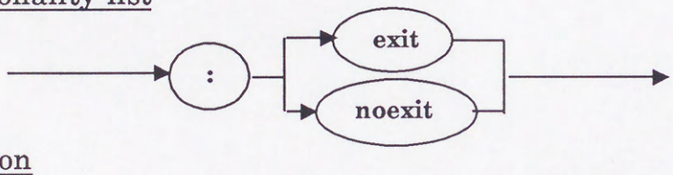
identifier-declarations



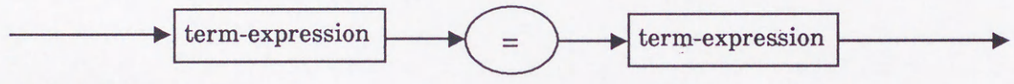
identifier-declaration



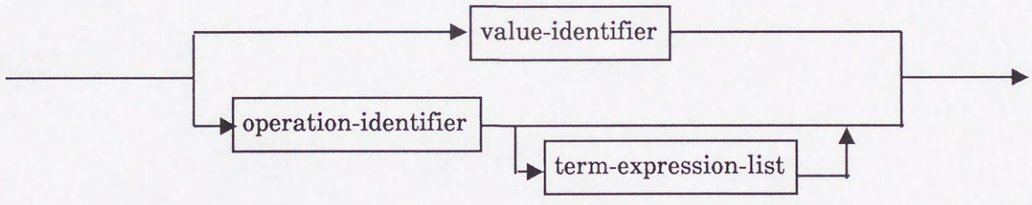
functionality-list



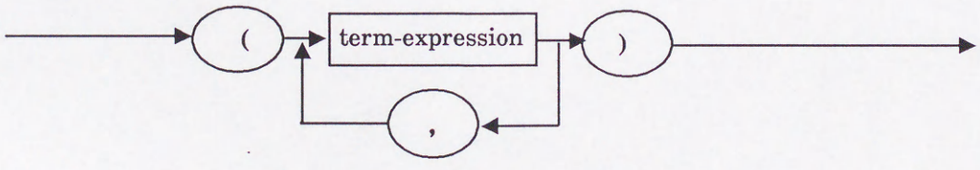
equation



term-expression

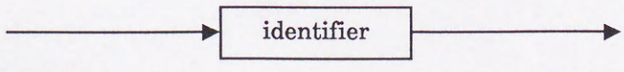


term-expression-list

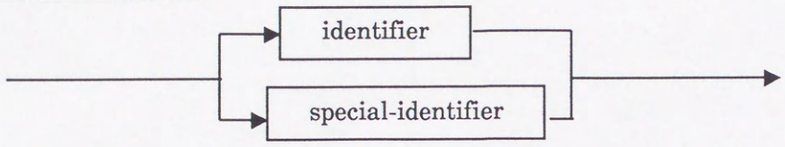


specification-identifier , type-identifier , sort-identifier ,

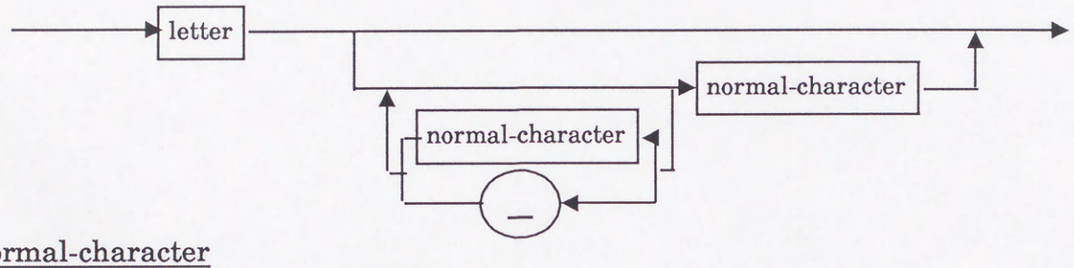
process-identifier , gate-identifier , value-identifier



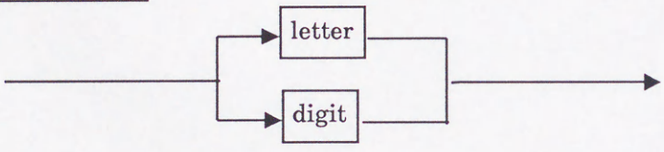
operation-identifier



identifier　(予約語と同じスペルを持ってはならない)



normal-character



letter

"a" - "z" , "A" - "Z"

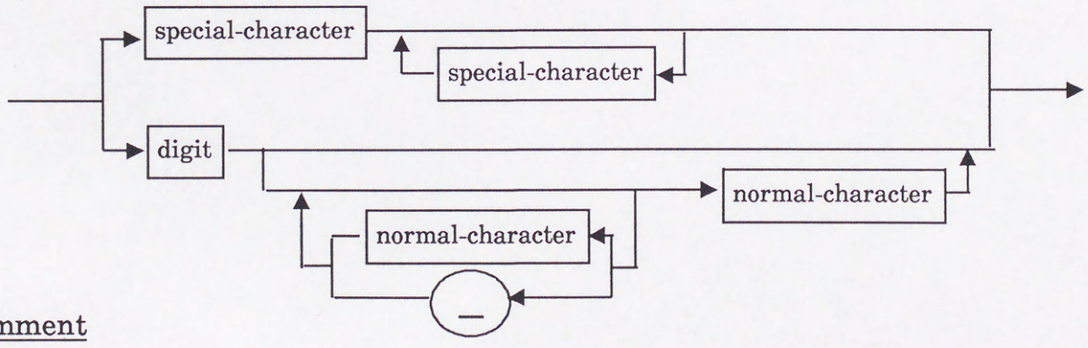
digit

"0" - "9"

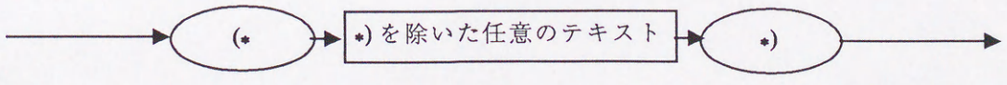
special-character

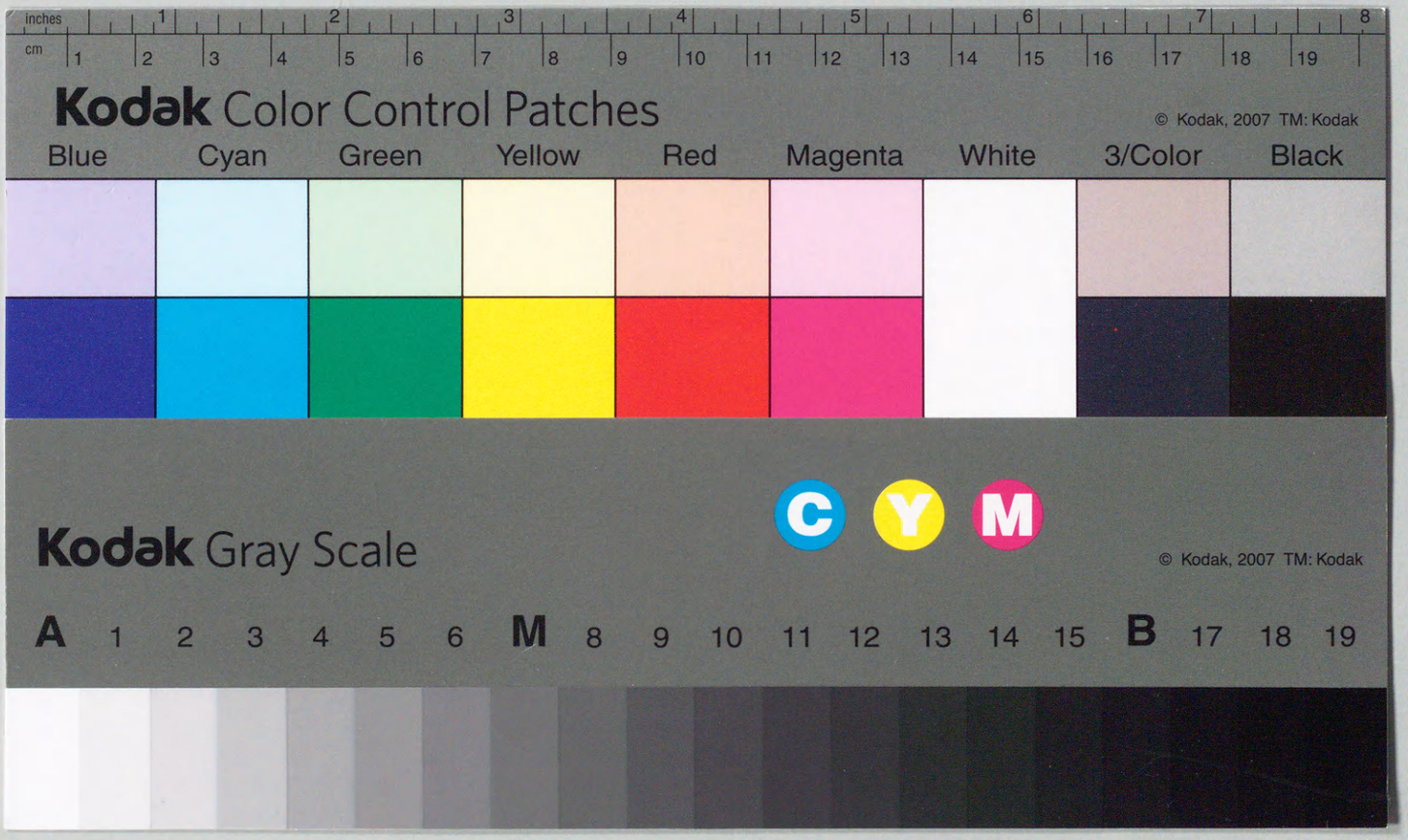
"*","+","-", "/","<", ">"

special-identifier　(予約語と同じスペルを持ってはならない)



comment




inches
cm
Kodak Color Control Patches
© Kodak, 2007 TM: Kodak
Blue
Cyan
Green
Yellow
Red
Magenta
White
3/Color
Black
Kodak Gray Scale
C
Y
M
© Kodak, 2007 TM: Kodak
A 1 2 3 4 5 6 M 8 9 10 11 12 13 14 15 B 17 18 19