形式手法による通信ソフトウェア試験データの生成とその試験法

辻 宏郷[†] 佐藤文明[†] 勝山光太郎[†]
水野忠則[†*] 曽我正和[†]

通信ソフトウェアのプロトコル規格に対する適合性を確認する手段として，ブラックボックス法に基づいた試験方法が通常用いられている．この場合，プロトコルが複雑になるにつれて，試験データの仕様を作成し，その仕様をもとに試験を行うことが困難となり，通信ソフトウェアの開発に要するコストや時間を増大させる原因となっている．我々は，この問題点を解決するために，形式手法に基づき，通信プロトコル試験データの仕様を記述するための専用言語 PSL. 1 を開発した．PSL. 1 を用いることによって，既存の試験記述法である TTCN では困難であった試験データ仕様におけるパラメータの検査箇所や検査範囲を形式的に記述可能となった．さらに，この PSL. 1 で記述された試験データ仕様をもとに，試験データの生成とその試験を行うための，試験データ生成ツール tdgen および試験データ解析ツール tdana を開発した．これらのツールを利用することによって，プロトコル仕様で決められている抽象的な仕様から実際に試験に用いる試験データを，試験の網羅性を考慮しつつ自動的に作成し，かつ通信ソフトウェアの適合性判定を自動的に行うことが可能となった．ここで示した方法を OSI 分散トランザクション処理に適用し，実際に適用可能であることを実証した．

Test Data Generation for Communication Software and Its Testing Method by Formal Approach

Hirosato Tsuji,[†] Fumiaki Sato,[†] Kotaro Katsuyama[†]
Tadanori Mizuno[†*] and Masakazu Soga[†]

The "black box testing" methodology is usually applied to the communication software testing in order to test the conformance to the protocol specification. However, as the complexity of protocols increases, so does the cost of the realization testing, since the description of test data specification and the realization of such specification are difficult. For the purpose of solving this issue, we applied the formal methods to the specification and the validation of the test data of the communication softwate testing. At first, we have developed the Parameter Specification Language PSL. 1 to describe the specification of the test data formally. And we have also developed test data generation tools which realize the automatic test data generation and validation from such formal specifications. Finally, we have applied our methodology to the testing of the OSI distributed transaction processing software, and proved the possibility of the application.

1. はじめに

ISO（国際標準化機構）における OSI (Open Systems Interconnection) の標準化が進展した結果，分散トランザクション処理[4]・ディレクトリ[5]といった OSI プロトコルの実装が各地で行われている．このような実装ソフトウェアにとって，規格に対する適合性を確認することは不可欠であり，このための試験に要するコストや時間を削減することが重要な課題となっている．通信ソフトウェアの開発においては機能試験・ヒートラン試験・性能試験等のさまざまな試験を実施するが，階層構成のエンティティを実装したプログラムの規格に対する適合性を確認する手段としては，ブラックボックス試験法に基づいた試験方法が通常用いられている[9]．**図1**は，ブラックボックス試験法に基づく試験形態の一例を示したものである．この方法では，試験対象 (IUT : Implementation Under Test) エンティティの実装プログラムを，上位テスタ (UT : Upper Tester) と下位テスタ (LT : Lower Tester) から構成された，試験システムと接続することによって試験を行う．そして，試験システムと試験対象エン

† 三菱電機(株)情報電子研究所
Computer and Information Systems Laboratory, Mitsubishi Electric Corporation
* 現在 静岡大学工学部情報知識工学科
Presently with Department of Computer Science, Faculty of Engineering, Shizuoka University

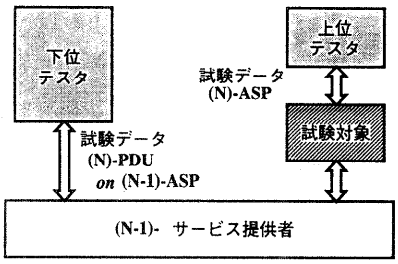


図 1 ブラックボックス試験法の試験形態
Fig. 1 Black Box Testing of communication software.

ティティとの間で交換する抽象サービスプリミティブ (ASP : Abstract Service Primitive) やプロトコルデータ単位 (PDU : Protocol Data Unit) を観測することによって，試験対象プログラムの振るまいを試験する．OSI 適合性試験法[1]は，この試験方法の典型的な一例である．

この試験方式はプロトコル誤りを検出するために有効であるが，試験を実施するためのコストが大きい[11]．第一に，試験システムと試験対象との間でやりとりする試験仕様（試験スィートや試験データ）の作成およびその試験に要するコストである．第二に，試験システムの開発に要するコストである．特に，下位テスタは通信相手を模倣した動作をする必要があるので，自動的に試験を実施する試験システムを開発することは容易ではない．さらに，OSI 応用層エンティティの複雑なプロトコルでは，上位テスタや下位テスタの動作によって試験対象の振るまいを制御不可能な場合があるため，完全なブラックボックスと見なした試験は困難であるという問題点がある．したがって，この試験方法を適用するためには，試験実施者は PICS (Protocol Implementation Conformance Statement) や PIXIT (Protocol Implementation eXtra Information for Testing) といった内部実装情報を入手する必要がある．あるいは，このような内部情報を必要としない試験システムを実現するためには，テスタ内部に試験対象に依存した専用処理機能を組み込まなければならない．現状では，内部実装情報を明らかにして自動的に試験を行うか，あるいは，試験実施者が通信相手を模倣して対話的に試験を行う形態が実施されている．したがって，試験対象プロトコルの複雑化とともに増加する試験コストを，削減する工夫が重要となっている．

本論文では，この問題を解決するために，試験仕様，特に試験データにおける検査項目や検査条件の仕様記述に，形式的手法を導入する．一般に，仕様記述に形式的手法を用いる目的は，

1. 標準に規定された非形式的な仕様定義から，一意に解釈可能な曖昧でない仕様定義の獲得
2. 形式仕様から実装ソフトウェアの自動生成
3. 設計仕様から試験仕様の自動生成

等がある (**図 2**)[15]．本稿では，通信ソフトウェアの中で特に OSI 応用層プロトコルの試験を対象とし，形式手法に基づいた試験データの生成とその試験法を示す．OSI 応用層ならびにプレゼンテーション層プロトコルの ASP や PDU のデータ構造は，ISO によって標準化された抽象構文記法 ASN. 1 (Abstract Syntax Notation One)[2),3)] を用いて定義されている．そこで，ASN. 1 を用いてデータ構造を規定された試験データに対して，検査項目や検査条件の仕様を形式的に記述するための，仕様記述言語 PSL. 1 を開発した．さらに，ASN. 1 と PSL. 1 を組み合わせて記述した試験データ仕様をもとに，試験対象と試験システムとの間でやりとりする試験データを，試験の網羅性を考慮しつつ自動的に作成する試験データ生成／解析ツールを開発した．最後に，これらを OSI 分散トランザクション処理の試験に適用し，実際に適用可能であることを実証した．

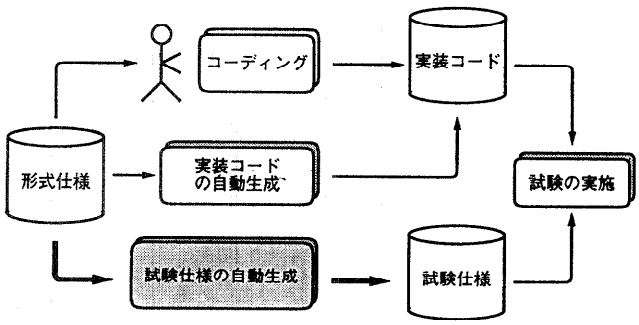


図 2 形式仕様に基づく実装と試験
Fig. 2 Implementation and testing from formal specifications.

以下，第 2 章では，OSI 通信ソフトウェアに対する試験における，試験データに関連するさまざまな問題を明らかにする．第 3 章では，これらの問題を解決する試験データ仕様記述言語 PSL. 1 について述べ，第 4 章では，試験データ生成／解析ツールについて詳述する．第 5 章では，これらを OSI 通信ソフトウェアの試験に対して適用した結果を報告する．最後に，第 6 章でまとめと今後の課題について述べる．

2. 通信ソフトウェア試験データに関する諸問題

2.1 ブラックボックス試験法と試験データ

OSI 適合性試験法の定義によれば，試験スィートは階層的な構造を持っている（**図3**）．すなわち，試験スィートは試験ケースの集合であり，試験ケースは試験イベントの順序列である．試験イベントは，ASP または PDU の，送信動作または受信動作である．本論文では，個々の試験イベントにおいて，試験対象と上位テスタ・下位テスタの間で交換されるべき ASP や PDU を，「試験データ」と定義する．ブラックボックス法に基づくプロトコルの適合性試験は，2種類の段階に分類することができる[8]．第一の段階は構造試験と呼ばれるもので，試験対象に対して試験イベントの順序列を与えることによって，入出力系列が試験ケース通りであることを確認する方法である．この試験方法では，個々の試験イベントにおける試験データ，およびその内部パラメータ値については考慮されない．構造試験において用いる試験ケースについては，形式的に記述されたプロトコル仕様から自動生成する手法が，多くの研究成果として報告されている[6),13),17)]．例えば，状態遷移機械に基づく形式記述技法（SDL）を用いて記述されたプロトコル仕様に対して，一定のアルゴリズムを適用することによって，TTCN で記述された試験ケースを自動的に作成することができる．第二の段階は機能試験と呼ばれるもので，プロトコル仕様で規定されたすべてのパラメータ値について，実装されていることを確認する方法である．応用層プロトコルのように試験対象プロトコルが複雑になるにつれて，ASP や PDU に含まれるパラメータ値の意味が重要となっている．例えば，意味的に異なる種類のサービスが一つの PDU で定義されており，その PDU に含まれた一要素のパラメータ値によってサービスの意味を区別する場合がある．したがって，限定した機能を試験する構造試験だけでは，プロトコルに対する適合性を確認するのに不十分であり，試験データのパラメータレベルまで試験する必要性がある．

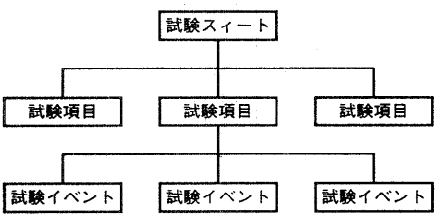


図 3 試験スィートの構造
Fig. 3 Test suite structure.

2.2 試験データの仕様化に関する問題

ブラックボックス試験に必要な試験仕様の作成においては，試験スィートに含まれるすべての試験データを仕様化する必要がある．本節においては，この試験データ仕様を作成する際の問題点について述べる．第一の問題点は，試験データは試験ケースの構成要素であるため，試験ケースに矛盾しないように試験データのパラメータを決定する必要性である．前節で述べたように，ASP や PDU のパラメータ値によっては意味が異なる場合があり，適切な値を設定しないとその試験ケースが成立し得ない．したがって，試験ケースを自動生成することができても，試験実施に必要な値を設定した試験データを作成することが，この試験ケースを用いるために不可欠となる．第二の問題点は，試験範囲が全体でない試験データの場合に，部分的なパラメータに対する試験仕様を記述する必要性である．通信ソフトウェアに対する試験においては，試験すべき箇所は試験データ全体とは限らず，その一部のパラメータを試験すればよい場合がある．さらに，これらのパラメータ値は一定の値を取る必要はなく，ある集合の中のいずれかの値であれば十分であったり，値が存在しさえすればその値は何でもよい場合がある．第三の問題点は，このような性質を持った試験データの仕様を形式的に記述する必要性である．OSI 標準規格において，このパラメータ値に関する情報は，自然言語を用いて非形式的に記述されているために曖昧である．また，計算機処理に都合の良い形式でないため，試験データ仕様から試験データの自動生成に使用することができない．それゆえに，試験データ仕様すなわちパラメータ設定条件を，形式的に記述する手法が必要である．

2.3 試験データの生成と試験法に関する問題

通信ソフトウェアの試験で使用する試験データは，下記の2種類に分類することができる．一方は，試験システムが試験対象に対して発行する ASP や PDU である．他方は，試験対象から試験システムが受信するべき ASP や PDU である．通信ソフトウェアの試験を実際に行うためには，抽象的な試験仕様として記述された試験データを実現することが必要である．すなわち，試験対象に対して発行する試験データの生成，ならびに受信した試験データの妥当性試験が必要

である．本節においては，試験データの実現時における問題点について述べる．第一の問題点は，試験対象に対して発行する試験データを生成する際の，すべてのパラメータ値を決定する必要性である．試験仕様は試験すべき箇所のパラメータ値やその範囲のみを記述するものであり，実際に試験対象との間でやりとりする試験データを生成するためには，試験実施者がその他のパラメータ値を決定する必要がある．第二の問題点は，試験対象から受信した試験データの試験方法である．試験箇所は試験データのパラメータ値全体とは限らないので，受信データを解析して試験仕様に記述された項目のみを試験する必要がある．それゆえに，受信することが期待される試験データを事前に作成しておき，受信データと試験データの単純なバイナリー比較を行うことはできない．第三の問題点は，試験対象の振るまいに応じた試験データを作成する必要性である．応用層プロトコルのような複雑なプロトコルの場合，試験対象の振るまいすべてを，上位テスタや下位テスタからの制御によって決定できない．この時，試験対象に発行する試験データのパラメータの一部は，試験対象からの受信データに含まれるパラメータ値を考慮して決定する必要がある．それゆえに，試験対象に発行する試験データすべてを事前に作成しておくことはできない．このような問題点が存在するため，試験対象プロトコルが変わるたびに，試験システム（上位テスタ・下位テスタ）内部に，試験データの生成や解析のための専用処理モジュールを用意しなければならない．このことが，試験システムを実装するために必要な開発期間を増大させる要因となっている．

3. 試験データ仕様記述用言語 PSL.1

3.1 既存の試験データ仕様記述法

前章で述べたように，試験データ仕様を曖昧なく記述して試験データの自動生成やその試験を行うためには，その仕様を形式的に記述する方法が必要である．本節では，ASN.1 型記法[2]を用いてデータ構造を定義された試験データに対する既存の仕様記述方法である，ASN.1 値記法[2]および TTCN[1]の問題点を示す．**図4**は，国際標準において ASN.1 型記法を用いて定義された，OSI 分散トランザクション処理の PDU の一例である．一般に，試験データに含まれるパラメータは，

（a） 特定の値であるべきパラメータ

（b） ある種の値集合に含まれる値のいずれかであ

```
TP-BEGIN-DIALOGUE-RI ::= SEQUENCE {
    CHOICE {
        dialogue [1] SEQUENCE {
            initiating-tpsu-title       [1] TPSU-TITLE OPTIONAL    ,
            recipient-tpsu-title        [2] TPSU-TITLE OPTIONAL    ,
            functional-units            [3] FU-list DEFAULT {
                                              shared-control ,
                                              commit-and-chained-trransactions
                                            }                      ,
            begin-transaction           [4] BOOLEAN OPTIONAL       ,
            confirmation                [5] ENUMERATED
                                              { always   (1),
                                                negative (2),
                                              } DEFAULT negative   ,
            dialogue-correlator         [6] Dialogue-channel-correlator ,
            last-partner-identifier     [7] Dialogue-channel-correlator
                                              OPTIONAL             ,
            user-data                   [8] User-information OPTIONAL
        } ,
        channel  [2] SEQUENCE {
            selected-FUs                [1] FU-list DEFAULT { recovery } ,
            channel-correlator          [2] Dialogue-channel-correlator  ,
            channel-utilization         [3] ENUMERATED
                                              { tppm-recovery (1)
                                                two-way-recovery (2)
                                              } DEFAULT two-way-recovery ,
            last-partner-identifier     [4] Dialogue-channel-correlator
                                              OPTIONAL
        }
    }
}
```

図 4 ASN.1 型記法を用いた PDU の定義例
Fig. 4 PDU definition using ASN.1 type notation.

るべきパラメータ

(c) 任意の値であってよいパラメータ

の三通りに分類することができ，このような条件を形式的に記述できることが望まれる．ASN.1 値記法は，試験データ仕様を形式的に記述する方法の一つであるが，この時はすべてのパラメータに対して具体的な値を記述しなければならない．したがって，上記の分類(b)や(c)に属するパラメータに対しても，すべての値を決定した上で仕様化を行うことを必要とし，結果として試験準備に要する作業時間を増加させることになる．さらに，試験データにおける検査箇所や検査条件が曖昧となる，といった欠点を生じることになる．

```
{ tp-begin-dialogue-ri {
        dialogue {
              initiating-tpsu-title          2          ,
              recipient-tpsu-title           3          ,
              functional-units               '011000'B  ,
              begin-transaction              true       ;
              confirmation                   always     ,
              dialogue-correlator            0
        }
    }
}
```

図 5 ASN.1 値記法を用いた PDU の記述例
Fig. 5 PDU description using ASN.1 value notation.

ASN.1 PDU Type Definition
PDU Type : TP-BEGIN-DIALOGUE-RI **PCO Type :** **Comment :**
Type Definition

SEQUENCE {
  CHOICE {
  -- ここに **ASN.1 値記法を用いた PDU 定義を引用する。**
  }
}

ASN.1 PDU Constraint Declaration
Constraint Name : TP-BD-RI-001 **PDU Type : TP-BEGIN-DIALOGUE-RI** **Derivation Path :** **Comments :**
Constraint Value

{
  dialogue {
  initiating-tpsu-title 2,
  recipient-tpsu-title 3,
  functional-units '011000'B,
  begin-transaction true,
  confirmation always,
  dialogue-correlator ? -- **任意の値でよい。**
  }
}

Detailed Comments : **dialogue-correlator の値は TP-BEGIN-DIALOGUE-RC に利用する。**

図 6 TTCN.GR を用いた試験データ仕様の記述例
Fig. 6 Test data specification using TTCN.GR.

図 5 は，ASN.1 値記法を用いて図 4 の PDU の試験データ仕様を記述した例であり，非試験項目である dialogue-correlator のパラメータ値 0 を記述する必要を生じている．一方，抽象試験仕様を記述するための言語 TTCN においても，前述の部分的なパラメータ仕様記述を可能とするための拡張が行われている．しかし，TTCN で試験仕様を記述するためには，ASN.1 型記法で定義された ASP や PDU を，TTCN の ASN.1 ASP/PDU Type Definition を用いて記述し直す必要がある．また，試験条件は ASN.1 ASP/PDU Constraint Declaration を用いて記述するが，Sequence 型のように構造をもった型に対しては，その型ごとに定義および制約の記述が必要となる．さらに，この制約はコメントとして記述可能であるため，自然言語を用いた非形式的な記述を許している．したがって，TTCN を用いた場合は記述量が膨大かつ曖昧になり，試験データ生成やその試験を自動化することが困難となる．

図 6 は，図 5 と同等の試験データ仕様を TTCN を用いて記述した例であり，ASN.1 PDU Type Definition の Type Definition の領域には，図 4 に示した PDU 定義をそのまま複写する必要を生じている．

3.2 ASN.1 パラメータ設定時の非決定性

我々は，ASN.1 型記法を用いた ASP や PDU のデータ構造定義を流用しつつ，試験に必要なパラメータ設定条件のみを，簡潔かつ形式的に記述可能な試験データ仕様記述用言語を開発した．このため言語仕様の設計にあたって，仕様記述言語に要求される記述能力を明確化した[12]．ASN.1 型記法を用いたデータ構造定義に対して，値をもった ASP や PDU を作成するためには，パラメータ値等の設定を必要とする．本論文では，この時必要となる設定を「ASN.1 パラメータの非決定性」と定義する．この非決定性は，以下の 4 通りに分類することができる．

```
psl-descriptions        ::=   psl-description psl-descriptions
                          |   psl-description .

psl-description         ::=   choicetype-condition
                          |   sequenceoftype-conditon
                          |   setoftype-conditon
                          |   sequencetype-conditon
                          |   settype-conditon
                          |   booleantype-condition
                          |   integertype-condition
                          |   enumeratedtype-condition
                          |   nulltype-condition
                          |   objectidentifiertype-condition
                          |   bitstringtype-condition
                          |   othertype-condition
                          |   comment .

choicetype-condition    ::=   "choice" path-name           ";"
                          |   "choice" path-name "=" "OPT" ";" .

sequenceoftype-conditon ::=   "seqof" path-name "=" seqof-number ";"
                          |   "seqof" path-name "=" "OPT"        ";" .

setoftype-conditon      ::=   "setof" path-name "=" seqof-number ";"
                          |   "setof" path-name "=" "OPT"        ";" .

sequencetype-conditon   ::=   "seq" path-name "=" "OPT" ";" .

settype-conditon        ::=   "set" path-name "=" "OPT" ";" .

booleantype-condition   ::=   "bool" path-name "=" true-value  ";"
                          |   "bool" path-name "=" false-value ";"
                          |   "bool" path-name "=" "OPT"       ";" .

integertype-condition   ::=   "int" path-name sign integer-number ";"
                          |   "int" path-name "=" "OPT"           ";" .

enumeratedtype-condition
                        ::=   "enum" path-name sign integer-number ";"
                          |   "enum" path-name "=" "OPT"           ";" .

nulltype-condition      ::=   "null" path-name "=" zero  ";"
                          |   "null" path-name "=" "OPT" ";" .

objectidentifiertype-condition
                        ::=   "objid" path-name "=" object-identifier ";"
                          |   "objid" path-name "=" "OPT"             ";" .

bitstringtype-condition ::=   "bit" path-name "=" bit-strings ";"
                          |   "bit" path-name "=" "OPT"       ";" .

othertype-condition     ::=   type-name path-name "=" octet-strings ";"
                          |   type-name path-name "=" "OPT"         ";" .

comment                 ::=   "#" any-strings .

type-name               ::=   any-strings .

path-name               ::=   "(" node-names ")" .

node-names              ::=   node-name "/" node-names
                          |   node-name .

node-name               ::=   any-strings .

integer-number          ::=   natural-number
                          |   zero
                          |   "-" natural-number .

zero                    ::=   "0" .

seqof-number            ::=   natural-number
                          |   zero .

true-value              ::=   "true" | "TRUE" .

false-value             ::=   "false" | "FALSE" .

sign                    ::=   "=" | "<" | ">" | "<=" | ">=" .

object-identifier       ::=   '"' object-identifier-value '"'
                          |   object-identifier-value .

object-identifier-value ::=   natural-number-or-zero object-identifier-value
                          |   natural-number-or-zero .

nutural-number-or-zero  ::=   seqof-number .

bit-strings             ::=   '"' bit-strings-value '"'
                          |   bit-strings-value .

bit-strings-value       ::=   bit-number bit-strings-value
                          |   bit-number .

bit-number              ::=   "0" | "1" .
```

図 7 PSL.1 構文の形式的定義
Fig. 7 Formal syntax of PSL.1.

（1） ある型のパラメータ値
（2） 省略可能な型 (*Optional* 型) に対するパラメータの設定／省略
（3） 単一な型の順列／集合で表現される型 (*SequenceOf* 型, *SetOf* 型) の要素の個数
（4） 複数の要素の中から一つの要素を選択する型 (*Choice* 型) の要素の選択

分類(1)に属する非決定性に必要となる情報の性質は，(1a)特定の値，(1b)ある値集合に含まれる値のいずれか，(1c)どのような値でもよい，のいずれかのケースに分類することができる．また，分類(2)に属する非決定性の場合は，(2a)常に値を設定する，(2b)常に値を省略する，(2c)どちらでもよい，のいずれかとなる．さらに，分類(3)に属する非決定性については，正の整数に対する(1)と同等であり，分類(4)に属する非決定性については，有限個の値集合(すなわち*Enumerated*型）に対する(1)と同等と見なすことができる．

3.3 PSL.1 の構文と意味

前節で述べた ASN.1 パラメータの非決定性に必要な情報を記述する言語として，PSL.1(Parameter Specification Language for ASN.1) を開発した．PSL.1は，試験データ仕様において非決定性を生じる各パラメータの試験条件を，パラメータごとに記述可能とすることを目的としている．本節では，PSL.1 の構文と意味を説明する．PSL.1 において使用可能な全構文の形式表現を**図7**に示す．試験データ仕様における各パラメータの識別は，非決定性の種類を表す ASN.1 の型名と，構造型のメンバ名を“／”を区切り記号をして列挙するパス名形式を，組み合わせることによって指定する．また，試験データ仕様として必要となるパラメータ値の条件を，等号・不等号ならびに値を用いて記述する．すなわち，*type* 型であるパス *path* のパラメータが値 *value* であることを要求する試験データ仕様を，

type(*path*)=*value*;

と記述する．前節で述べた分類(1)の非決定性に対して(1a)に属するパラメータの設定条件，例えば，整数型である dialogue-correlator が0であることを，

int(tp-begin-dialogue-ri/dialogue/dialogue-correlator)=0;

と記述する．また，数値を取り得る型に対しては，不等号 "<", ">" 等を用いることで，分類(1b)に属する設定条件を記述する．例えば，列挙型である provider-diagnostic が1以上8以下であることを，

enum(tp-begin-dialogue-rc/dialogue/provider-diagnostic)>=1;
enum(tp-begin-dialogue-rc/dialogue/provider-diagnostic)<=8;

のように複数条件式の積を用いて記述する．分類(1c)に属するパラメータについては，そのパラメータに対する条件式を記述しないことによって，任意の値を取り得ることを意味する．また，分類(2)の非決定性に対するパラメータの省略を明示的に記述する場合は，値の代わりに予約語 OPT を用いて記述する．例えば，ビット列型である functional-units を省略することを，

bit(tp-begin-dialogue-ri/dialogue/functional-units)=OPT;

と記述する．さらに，分類(3)に対する要素の個数は，

seqof(tp-abort-ri/user/user-data)=3;

と記述し，分類(4)に対する要素の選択は，

choice(tp-end-dialogue-ri);

と記述する．ところで，文字列で表現可能なパラメータ (*PrintableString* 型) やオクテット列で表現可能なパラメータ (*OctetString* 型) の場合は，その取り得るべきパラメータ値を等号 "=" を用いて記述する．例えば，オクテット列 password が "beatrice" という文字列であることを，

octet(credentials/simple/password)="beatrice";

と記述する．一般に，階層構造における上位エンティティの PDU は，下位エンティティの ASP の一要素にマッピングされる．そこで，PSL. 1はこのような文字列・オクテット列のパラメータ値を，外部ファイルに格納された値を参照する形式で指定する機能を提供する．例えば，user-data にマッピングすべき PDU が "U-ASE-DATA" というファイルに格納されていることを，

any(tp-begin-dialogue-ri/dialogue/user-data)<"U-ASE-DATA";

と記述する．この時の記号 "<" の意味は不等号ではなく，読み込むファイルを指定するリダイレクション記号である．**図8**は，PSL. 1 を用いて図5・図6と同等の試験データ仕様を記述した例である．従来の記述法と比較して，PSL. 1 では曖昧さがなくなり試験範囲が明確化している反面，パラメータの型名を記述する必要がある．この型名は非決定性を識別するためには冗長な情報であるが，仕様の理解性向上を目的として付加している．なお，後述するツールを利用して対話的に PSL. 1 記述を作成可能であるため，PSL. 1 記述作成者は，この型情報に関する知識を必要としない．

```
choice(tp-begin-dialogue-ri) ;
choice(tp-begin-dialogue-ri/dialogue) ;
int(tp-begin-dialogue-ri/dialogue/initiating-tpsu-title) = 2 ;
int(tp-begin-dialogue-ri/dialogue/recipient-tpsu-title) = 3 ;
bit(tp-begin-dialogue-ri/dialogue/functional-units) = "011000" ;
bool(tp-begin-dialogue-ri/dialogue/begin-transaction) = true ;
enum(tp-begin-dialogue-ri/dialogue/confirmation) = 1 ;
int(tp-begin-dialogue-ri/dialogue/last-partner-identifier) = OPT ;
seqof(tp-begin-dialogue-ri/dialogue/user-data) = OPT ;
```

図8 PSL. 1 を用いた試験データ仕様の記述例
Fig. 8 Test data specification using PSL. 1.

4. 試験データ生成／解析ツールの開発

4.1 試験データ生成／解析ツールの構成

我々は，前章にて述べた PSL. 1 を用いてパラメータ設定条件を記述した試験データ仕様に基づき，試験対象の振るまいに応じた試験データの自動生成や解析を実現するために，試験データ生成／解析ツールを開発した．このツールは，試験データ生成ツール tdgen と試験データ解析ツール tdana の二つのプログラムから構成されている（**図9**）．ASN. 1 型記法で記述された試験データの構造定義（一つの ASN. 1 型定義ファイル）は，両方のツールに共通に必要な入力ファイルである tdgen は，PSL. 1 で記述された試験データ仕様(任意の個数の PSL. 1 記述ファイル)を入力とし 非決定性パラメータの値をすべて決定した後，ASN. 1符号化規則[3]に基づいて符号化した試験データを生成する．一方 tdana は，符号化された試験データを入力とし，復号化を行った後，設定されていた非決定性パラメータの設定条件を PSL. 1 形式に逆変換する．**図10**に，tdgen

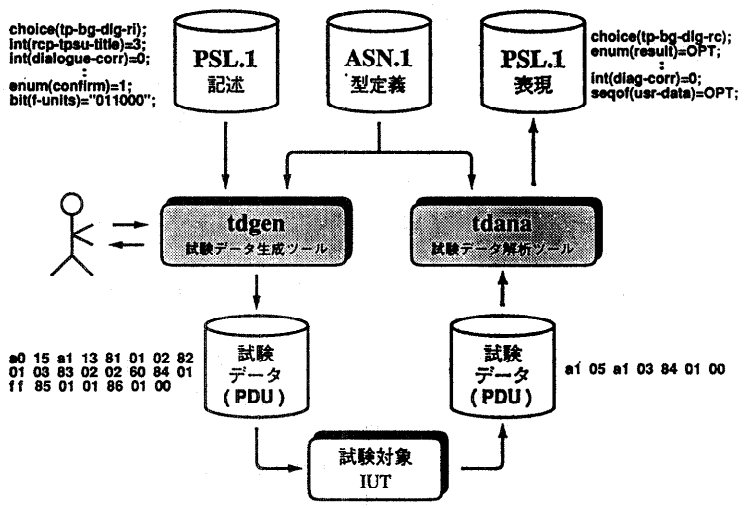


図 9 試験データ生成／解析ツールの構成
Fig. 9 Structure of test data generation tools.

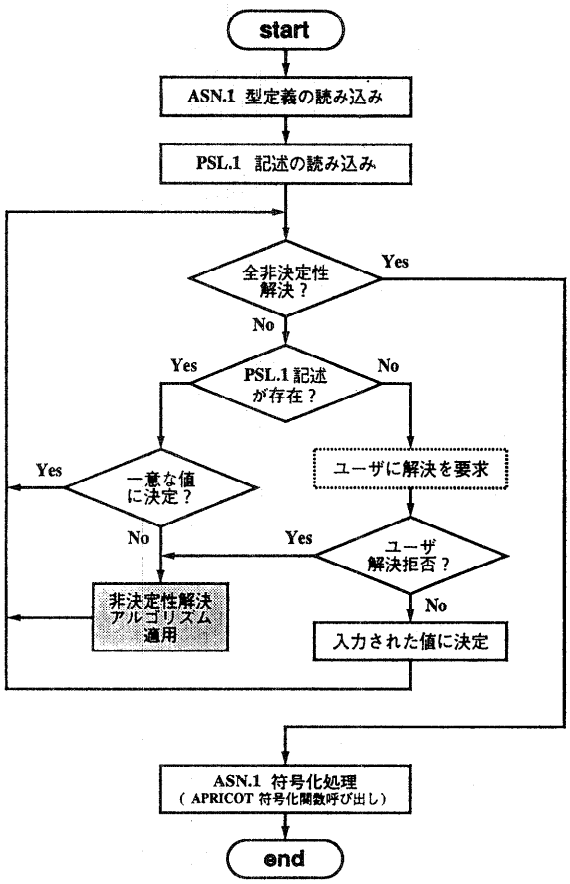


図 10 tdgen のアルゴリズム
Fig. 10 Algorithm of tdgen.

の主要アルゴリズムを示す．各プログラムはC言語を用いてコーディングを行い，ASN.1の符号化／復号化処理は，ASN.1 ツール APRICOT[18]のコーデックライブラリの符号化／復号化関数を呼び出すことによって実現した．

4.2 試験データ生成ツール tdgen

試験データ生成ツール tdgen は，パラメータ設定条件を記述した入力ファイルをもとに，すべてのパラメータ設定値を決定し，試験データを生成する．パラメータ設定に必要な情報が不足する場合は，ユーザとの対話入力によって補足したり，後述する非決定性解決アルゴリズムを用いて自動的に補足を行う．パラメータの設定値は，以下の優先順位に従って決定する．

1. PSL. 1 で記述された設定条件
2. ユーザとの対話的入力
3. 非決定性解決アルゴリズムを用いた自動設定

すなわち，パラメータ設定条件が PSL. 1 を用いて記述されている場合は，それらの条件式を満たすパラメータを設定する．非決定性を解決する設定条件が記述されていない時は，tdgen の二つの実行モードによって異なった処理を行う．対話モードで実行した場合は，tdgen は非決定性の解決（すなわち値の入力や選択）をユーザに要求する．この要求に対し，試験仕様として任意の値を取り得る場合は，ユーザは回答入力を拒否することができる．この時は，後述する非決定性解決アルゴリズムを用いてパラメータを自動的に決定する．**図 11** は，対話モードを用いて図5～図7の試験仕様を満たす試験データを作成した例である．一方，全自動モードにおいては，ユーザとの対話入力は行わない．すなわち，PSL. 1 で設定条件が記述されないすべてのパラメータ設定に対して，非決定性解決アルゴリズムを適用して試験データを自動生成する．**図 12** は，全自動モードを用いて試験データを作成した例である．試験仕様として最低限必要な条件を記述したファイル（図7）を入力として，その他のパラメータを自動的に決定している．上記以外の機能として，tdgen はユーザとの対話入力や自動決定で得られた非

```
% tdgen -tx > TP-BEGIN-DIALOGUE-RI.exs
=====================================
 ASN.1 Test Data Generator (1991/03)
=====================================
Input ASN.1 Table File Name ?        > TPASE-APDU-2ndDIS.t
Input ASN.1 Type/Path to Encode ?    >
Input PSL.1 Description File Name ?  >

ASN.1 Definition File Name      : "TPASE-APDU-2ndDIS.t"
ASN.1 Type to encode Test Data  : TPASE-APDU
ASN.1 Path to encode Test Data  :
Generation Mode No.             : 0101

[CHOICE] There are 22 choices at "".
        1) tp-begin-dialogue-ri
        2) tp-begin-dialogue-rc
                :
                :
        22) tp-association-establishment-rc
Please Input Node Number ? (1-22) > 1

[CHOICE] There are 2 choices at "tp-begin-dialogue-ri".
            1) dialogue
            2) channel
Please Input Node Number ? (1-2) > 1

[INT] tp-begin-dialogue-ri/dialogue/initiating-tpsu-title [OPT] ? > 2
[INT] tp-begin-dialogue-ri/dialogue/recipient-tpsu-title [OPT] ? > 3
[BIT] tp-begin-dialogue-ri/dialogue/functional-units [OPT] ? > 011000
[BOOL] tp-begin-dialogue-ri/dialogue/begin-transaction [OPT] ? > true
[ENUM] tp-begin-dialogue-ri/dialogue/confirmation [OPT] ? > 1
[INT] tp-begin-dialogue-ri/dialogue/dialogue-correlator ? > 0
[INT] tp-begin-dialogue-ri/dialogue/last-partner-identifier [OPT] ? > opt

[SEQOF] Path "tp-begin-dialogue-ri/dialogue/user-data" is OPTIONAL.
           Do you want to make This Node ? (y/n) > n

------------------------------
a0 15 a1 13 81 01 02 82 01 03 83 02 02 60 84 01
ff 85 01 01 86 01 00
```

図 11 OSI-TP PDU 対話型生成の実行例
Fig. 11 OSI-TP PDU interactive generation.

```
% tdgen -r TPASE-APDU-2ndDIS.t TP-BEGIN-DIALOGUE-RI TP-BEGIN-DIALOGUE-RI.exs
=====================================
 ASN.1 Test Data Generator (1991/03)
=====================================
ASN.1 Definition File Name      : "TPASE-APDU-2ndDIS.t"
ASN.1 Type to encode Test Data  : TPASE-APDU
ASN.1 Path to encode Test Data  :
Test Data File Name to Generate : "TP-BEGIN-DIALOGUE-RI"
PSL.1 Description File Name     : "TP-BEGIN-DIALOGUE-RI.exs"
Generation Mode No.             : 0040

----- PSL.1 Description -----
[CHOICE] "tp-begin-dialogue-ri" at ""
[CHOICE] "dialogue" at "tp-begin-dialogue-ri"
[SEQOF/SETOF] "tp-begin-dialogue-ri/dialogue/user-data" is  OPT
[INT] "tp-begin-dialogue-ri/dialogue/initiating-tpsu-title" = 2
[INT] "tp-begin-dialogue-ri/dialogue/recipient-tpsu-title" = 3
[BOOL] "tp-begin-dialogue-ri/dialogue/begin-transaction" = true
[ENUM] "tp-begin-dialogue-ri/dialogue/confirmation" = 1
[INT] "tp-begin-dialogue-ri/dialogue/last-partner-identifier" = OPT
[BIT] "tp-begin-dialogue-ri/dialogue/functional-units" = "011000"

------------------------------
tp-begin-dialogue-ri/dialogue/initiating-tpsu-title = 2 (INT)
tp-begin-dialogue-ri/dialogue/recipient-tpsu-title = 3 (INT)
tp-begin-dialogue-ri/dialogue/functional-units = 0110 (BIT)
tp-begin-dialogue-ri/dialogue/begin-transaction = true (BOOL)
tp-begin-dialogue-ri/dialogue/confirmation = 1 (ENUM)
tp-begin-dialogue-ri/dialogue/dialogue-correlator = 0 (INT)

------------------------------
a0 15 a1 13 81 01 02 82 01 03 83 02 02 60 84 01
ff 85 01 01 86 01 00
```

図 12 OSI-TP PDU 自動生成の実行例
Fig. 12 OSI-TP PDU automatic generation.

決定性に対する回答を PSL. 1 の構文形式で出力する機能があり，この出力テキストをファイルに記録することができる．このことは，テキストエディタ等を用いた編集作業を行うことなく，tdgen の入力である PSL. 1 記述ファイルを，tdgen 自身を用いて作成可能であることを意味する．図 11 の例では，対話的に試験データを作成すると同時に，図 7 に相当する PSL. 1 記述ファイルを作成している．さらに，tdgen は ASN. 1 型定義や PSL. 1 記述の解析のみを実行する機能を提供し，自らの入力となる試験データ仕様の作成を支援する[14]．

4.3 非決定性解決アルゴリズム

PSL. 1 を用いて記述された試験データ仕様のパラメータ設定条件が一つの値に決定できない条件の場合，あるいは設定条件が記述されていない場合には，tdgen は非決定性解決アルゴリズムを用いてパラメータ値を決定する．また，対話モードでユーザが回答を拒否した場合にも使用される．このアルゴリズムは，同値分割法 (equivalence partitioning) や境界値解析法 (boundary value analysis)[9] に基づいている．すなわち，条件式を成立する（同値分割における有効同値クラス）ための境界値の試験を必須とし，必要に応じて条件式を満たすその周辺値を試験する．この時のパラメータ値の決定方法を例を用いて示す．例えば，整数型に対するパラメータ設定条件が「5 以上 15 未満」の試験データ仕様に，非決定解決アルゴリズムを適用する．ユーザが最低レベルの網羅性，すなわち唯一つの試験データを要求している場合は，$x=5$ または $x=14$ のいずれかの値を設定する．通常レベルの網羅性に対しては，上記の二通りの試験データを作成する．また詳細レベルでは，上記の二値に加えて $x=6$〜13 の整数値のいずれか一つを乱数を用いて選択する[10]．なお，ASN. 1 で規定された型の値域（例えば整数型の場合 $-\infty$〜$+\infty$）と実装形態（32 ビットで表現可能な整数）とのギャップを埋めるために，非決定性解決アルゴリズムでは，これらの値域に制限を加えるための暗黙の設定条件に相当するコーディングを含んでいる．

4.4 試験データ解析ツール tdana

試験データ解析ツール tdana は，tdgen を用いて作成された試験データや試験対象から受信した試験データを復号化し，試験データに含まれるパラメータ解析結果を出力する．この時の解析情報（パラメータの値や選択など）をすべての非決定性に対する PSL. 1 構文形式の回答として出力すること，すなわち tdgen と逆の動作を行うことができる．したがって，この解析結果をファイルに保存することによって，tdgen の入力ファイルとして再利用することが可能である．tdgen と tdana を組み合わせて使用することによって，試験対象の振るまいに応じた試験データの作成や，試験すべき箇所のみに対する試験の実施が可能となる．具体的な方法については，次章にて詳述する．

5. OSI 通信ソフトウェアの試験への適用と評価

5.1 OSI 分散トランザクション処理への適用

OSI 分散トランザクション処理 (OSI-TP) は OSI 応用層プロトコルの一要素であり，通信ソフトウェアに対する試験における諸問題を示す例に適している．我々は，PSL. 1 および試験データ生成／解析ツールを評価するために，OSI-TP の試験に対して FOREST[16] と組み合わせて適用し，従来の試験手法との比較検討を実施した．FOREST は，OSI 通信ソフトウェアの開発を支援する試験システムであり，SDL を用いて記述された振るまい仕様から，試験ケースを自動生成することが可能となっている．**図 13** は，自動生成された試験ケースの一例である．試験ケースは FOREST システムが解釈を行い，個々の試験イベントにおける試験データの発行／受信，ならびに試験結果の判定を行う．試験適用の準備として，最初に TP プロトコルマシンの振るまいを，FOREST の入力仕様である SDL のサブセットを用いて記述した．また，ASN. 1 を用いて記述された TP の PDU 定義を，ASN. 1 プリコンパイラの入力構文を満たすように若干修正した上，同プリコンパイラへ入力することによって，試験データ生成／解析ツールの入力となる ASN. 1 型定義テーブルへと変換した．他方で，SDL で記述された動作仕様から自動生成された試験ケース（TTCN の behaviour description の構文で記述）を満たすために

```
U!    TP-BEGIN-DIALOGUE.req          # STATE = IDLE
L?    TP-BID-RI
L!    TP-BID-RC                      # STATE = AWBIDCNF
U?    nothing
L!    P-TOKEN-GIVE-RI                # STATE = AWPTGIND
L?    TP-BEGIN-DIALOGUE-RI
L!    P-TOKEN-PLEASE-RI              # STATE = DLGEST2
L?    P-TOKEN-GIVE-RI
U!    TP-END-DIALOGUE.req            # STATE = DLGEST
L?    TP-END-DIALOGUE-RI
```

図 13 OSI-TP 試験ケースの例
Fig. 13 Example of OSI-TP test case.

必要となる試験データ仕様，すなわちパラメータ設定情報をPSL.1を用いて記述した．

5.2 試験対象の振るまいに応じた試験データ生成の実現

OSI-TP の試験において，ダイアログを試験対象側から確立する試験シナリオを例として考える．試験対象から発行された TP-BEGIN-DIALOGUE-RI PDUに含まれる dialogue-correlator パラメータの値は，試験対象のプロトコルマシン内部で決定し，外部の試験システムから制御することが不可能である．そして，試験システムから試験対象への応答である TP-BEGIN-DIALOGUE-RC PDUを発行する際に，この PDU に含まれる dialogue-correlatorパラメータは同一の値を設定しなければならない．このため下位テスタには，受信した PDU に含まれていた値を保存し，試験対象に発行する PDU に同一の値を設定するメカニズムが必要となる．従来の方法では，試験実施者は dialogue-correlator に適当な値を設定した TP-BEGIN-DIALOGUE-RC を作成しておき，受信した TP-BEGIN-DIALOGUE-RI の dialogue-correlator の値を符号化箇所にコピーする特別なモジュールを下位テスタ内部に実装していた．この方法では，解析情報の再利用が必要となる度にテスタ内部の専用処理モジュールが必要となり，結果としてテスタの汎用性を低下させる原因となっていた．**図14**は，このような問題を解決するために，PSL.1 と試験データ生成／解析ツールを適用した例である．最初に，PSL.1で表現される試験データ解析結果から，応答用試験データ生成に必要な条件式を抽出する，条件式抽出モジュール (CEM : Condition Extraction Module) を実装する．tdgen では，複数のファイルに分割して記述した PSL.1 条件式を，入力として使用することができる．この性質を利用することによって，試験対象の振るまいの影響を受けない部分については，PSL.1 を用いてパラメータ設定条件を事前に記述しておく．試験実施に際しては，受信データを tdana を用いて PSL.1 表現に変換する（**段階1～段階3**）．

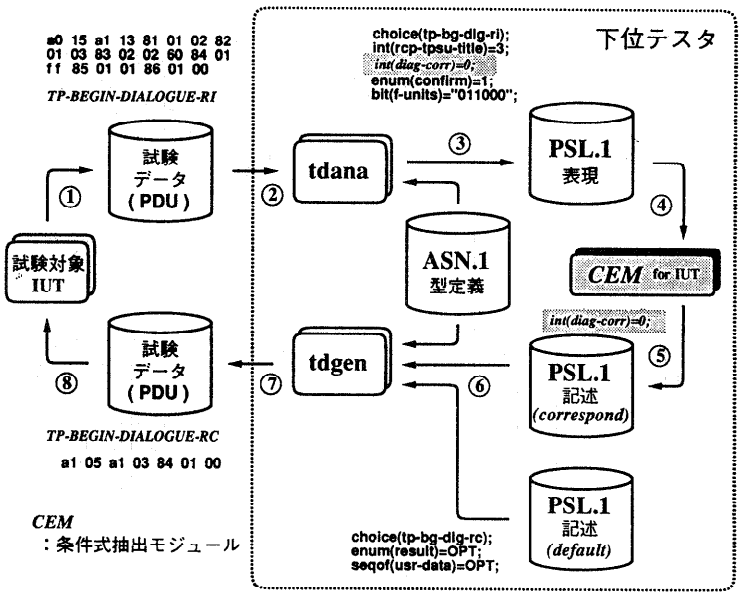


図 14 試験対象の動作に応じた試験データの生成
Fig. 14 Test data generation corresponded to IUT behaviour.

そして，応答用試験データ作成に必要となる PSL.1 記述を CEM を使用して切り出す（**段階4～段階5**）．その後，事前に記述しておいた PSL.1 記述と抽出された PSL.1 表現の組み合わせを tdgen の入力ファイルとして使用することによって，試験対象の振るまいに応じた試験データを即時生成し，試験対象に対して送信する（**段階6～段階8**）．本例において CEM は，

int(tp-begin-dialogue-ri/dialogue/dialogue-correlator)=…

という文字列を抽出し，“ri” を “rc” に置換することによって，

int(tp-begin-dialogue-rc/dialogue/dialogue-correlator)=…

という条件式に変換する．CEM が行う処理は試験対象に依存して変化するが，上記の例のように単純な文字列の検索・置換処理で済むので，既存のフィルタコマンド等を利用して簡単に作成することができる．

5.3 試験すべき箇所のみに対する試験の実現

試験対象から受信した試験データにおいては，すべてのパラメータが試験すべき項目であるとは限らないし，あるいは一定の値であることが要求されない場合がある．例えば，前述のダイアログ確立時の dialogue-correlator パラメータは，システム全体で一意であればよく，どのような値が送られてくるか全く予想が

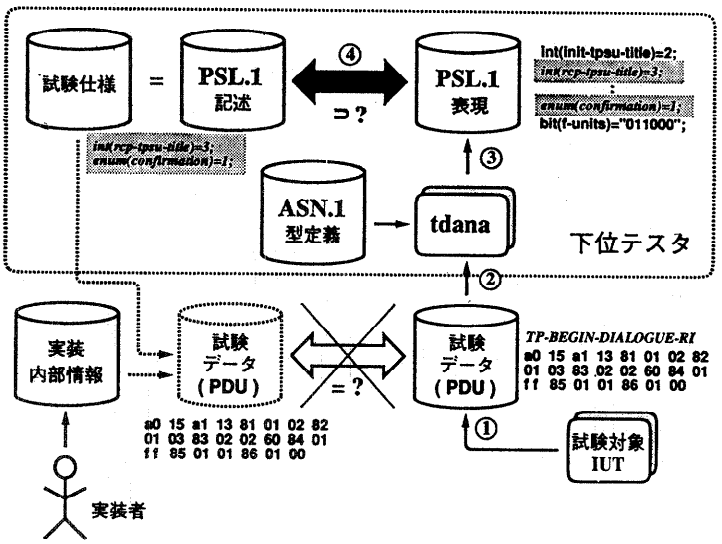


図 15 PSL.1 変換後の試験データ比較判定
Fig. 15 Validation of test data after PSL.1 conversion.

つかない．理想的な試験を考えるならば，受信したPDU を復号化した後に試験必要箇所のみを検査することが望ましい．しかしながら，この試験方法を実現するためには，下位テスタ内部にすべての受信 PDU に対する専用の検査モジュールが必要となる．従来の方法では，標準に規定されていない（実装者依存の）パラメータについては，試験実施者がプログラム実装者に内部情報を問い合わせることによって，受信することが期待されるすべてのパラメータ値を明らかにしていた．そして，受信した PDU と単純一致比較できるようなデータを事前に作成しておき，受信した試験データとの比較判定を行っていた．このためには実装内部情報の入手が不可欠であり，かつ試験すべきパラメータとそうでないパラメータの区別が曖昧になる，といった欠点をもっていた．**図 15** は，PSL.1 および試験データ解析ツールを利用することによって，試験すべき箇所のみに対する試験を実施した例である．まず最初に，試験対象から受信する試験データについては，試験すべき箇所のみに対する試験仕様を PSL.1 の構文を用いて記述しておく．試験対象から PDU を受信した時は，用意しておいた PDU と単純に比較する代わりに，tdana を用いて PSL.1 表現へと変換を行う（**段階 1～3**）．そして，試験システム内部で PSL.1 構文の式ごとに比較を行う（**段階 4**）．このように，試験対象箇所に対する必要条件すべてについて受信した試験データに含まれているか否かを確認することによって，試験対象箇所のみに対する試験が可能となった．

5.4 試験適用に対する評価

PSL.1 を用いて試験データ仕様を記述し，試験データ生成／解析ツールを用いて試験データの生成とその試験を行った．**表 1** は，ダイアログ確立時に交換する TP-BEGIN-DIALOGUE-RI/RC のパラメータを分類したものである．表において，指定どおりに設定されていることを確認する試験項目は分類（a）のみであり，また，分類（c）は試験対象の動作に依存する項目である．このように，試験データ仕様において特定の値であるべきパラメータは約 2/3 である．実際に試験に適用した際に作成した試験データにおいて，試験仕様として記述した PSL.1 記述と，すべての非決定性に対して記述を行った場合の比較結果を**表 2** に示す．試験項目のみを記述することによって試験範囲を明確化すると共に，すべてのパラメータ仕様を記述する必要のあった従来の試験と比較して，試験仕様の記述量を約 2/3 に削減することができた．また，この方法を適用することによって，実装者に問い合わせる必要があった内部の実装情報を不要とし，テスタ内部に作成していた例外処理を，単純な文字列検索・置換を行う CEM に集約することができた．

表 1 OSI-TP PDU パラメータの分類
Table 1 Classification of OSI-TP PDU parameter.

	PDU	
	TP-BEGIN-DIALOGUE-RI	TP-BEGIN-DIALOGUE-RC
(a)	initiating-tpsu-title recipient-tpsu-title functional-units begin-transaction confirmation user-data	result user-data
(b)	correlator last-partner-identifier	functional-units diagnostic
(c)		correlator

(a) サービスで指定し，外部から制御可能なパラメータ
(b) TP 内部で決定し，外部から制御不能なパラメータ
(c) 受信 PDU に応じて，設定値を変更すべきパラメータ

表 2 試験仕様記述量の比較
Table 2 Comparison of test specification description.

PDU		PSL. 1 記述		
名前	個数	(a)	(b)	(a)/(b)
TP-BEGIN-DIALOGUE-RI	16	180	270	66.6%
TP-BEGIN-DIALOGUE-RC	5	41	65	63.1%
TP-END-DIALOGUE-RI	2	11	16	68.8%
TP-ABORT-RI	4	21	36	58.3%
TP-INITIALIZE-RI	4	29	44	65.9%
合計	31	282	431	65.4%

(a) 実際に試験仕様として記述した条件式の数
(b) すべての非決定性を記述するのに必要な条件式の数

6. 結論

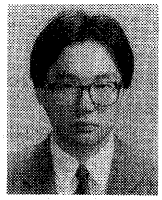
本論文では，通信ソフトウェアの試験において，試験対象プログラムと試験システムの間で交換する試験データの問題点を明らかにした．そして，試験データ仕様を形式的に記述するための仕様記述言語 PSL. 1 を開発した．また，PSL. 1 で記述された試験データ仕様に基づいて試験データの自動生成ならびに解析を実現する，試験データ生成／解析ツールを開発し，これらを OSI 分散トランザクション処理の試験に適用した．PSL. 1 を用いることによって，試験の実行に必要となる試験仕様を簡潔かつ厳密に記述することができた．また，試験データ生成ツール tdgen と試験データ解析ツール tdana を試験データの生成と解析に利用することによって，試験対象に依存してテスタ内部に必要となる，専用処理モジュールを削減することができた．この結果，試験仕様の記述量を減少するとともに，試験システムの汎用性を向上し，試験全体に要するコストを削減することができた．今回は PSL. 1 を試験データの仕様化に適用したが，今後は SDL を用いた状態遷移記述と組み合わせることによって，実装プログラムの自動生成への適用を検討していく．

謝辞 本論文の作成にあたり有益な助言をいただいた，モントリオール大学 Gergor v. Bochmann 博士ならびにビルケント大学 Behcet Sarikaya 博士に感謝の意を表します．

参考文献

1) ISO/IEC : OSI Conformance Testing Methodology and Framework, ISO 9646-1～5 (1991).
2) ISO/IEC : OSI—Specification of Abstract Syntax Notation One (ASN. 1), ISO 8824 (1990).
3) ISO/IEC : OSI—Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN. 1), ISO 8825 (1990).
4) ISO/IEC : OSI—Distributed Transaction Processing, ISO/DIS 10026-1. 2～3. 2 (1991).
5) ISO/IEC : OSI—The Directory, ISO/IEC 9594-1～8 (1990).
6) ISO/IEC : Working draft on Formal methods in Conformance testing, SC 21 N 7081 (1992).
7) CCITT : Functional Specification and Description Language, Recommendation Z. 100 (1988).
8) Holzmann, G. J. : *Design and Validation of Computer Porotocols*, Prentice Hall (1991).
9) Myers, G. J. : *The Art of Software Testing*, John Wiley & Sons (1979).
10) Pachl, J. : A Notation for Specifying Test Selection Criteria, *Protocol Specification, Testing and Verification X*, pp. 71-84, North-Holland (1990).
11) Eswara, S., Berriman, T., VanHoutte, P. and Sarikaya, B. : Towards Execution of TTCN Test Cases, *Protocol Specification, Testing and Verification X*, pp. 99-112, North-Holland (1990).
12) Sample, M. and Neufeld, G. : Support for ASN. 1 within a Protocol Testing Environment, *Third International Conference on Formal Description Techniques*, pp. 295-302, North-Holland (1990).
13) 岡崎，高橋，白鳥，野口：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，Vol. J 74-B-I, No. 10, pp. 733-747 (1991).
14) 長谷川，野村，堀内：ASN. 1 支援ツールの開発—コンパイラおよびエディター，情報処理学会マルチメディア通信と分散処理研究会資料，39-4 (1988).
15) Mizuno, T., Munemori, J., Sato, F., Nakakawaji, T. and Katsuyama, K. : COTTAGE : Systematic Method for the Development of Communication Software, *Protocol Specification, Testing and Verification VIII*, pp. 269-280, IFIP, North-Holland (1988).
16) 勝山，佐藤，中川路，水野：国際標準形式記述技法に基づく体系的な試験支援環境 FOREST の提案と実現，情報処理学会論文誌，Vol. 31, No. 7, pp. 1123-1133 (1990).
17) 佐藤，宗森，勝山，水野：通信システムの段階的な試験のための試験系列自動生成手法とその実現，情報処理学会論文誌，Vol. 31, No. 10, pp. 1486-1496 (1990).
18) 中川路，勝山，宮内，水野：OSI 抽象構文記法支援ソフトウェア APRICOT の開発と評価，電子情報通信学会論文誌，Vol. J 73-D-I, No. 2, pp. 225-234 (1990).

(平成 4 年 11 月 2 日受付)
(平成 5 年 4 月 8 日採録)

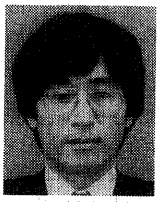


辻　宏郷

昭和 38 年生．昭和 63 年東北大学工学部情報工学科卒業．平成元年三菱電機(株)入社．現在同社情報電子研究所システム技術開発部．通信ソフトウェア開発環境，分散処理システムの研究開発に従事．分散環境における情報管理，セキュリティに興味を持つ．電子情報通信学会会員．

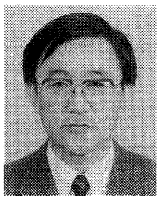
佐藤　文明（正会員）

昭和 37 年生．昭和 59 年岩手大学工学部電気工学科卒業．昭和 61 年東北大学大学院修士課程修了．同年三菱電機(株)入社．現在，同社情報電子研究所にて通信ソフトウェア開発環境，形式記述技法，並列分散処理システムへの研究開発に従事．工学博士．電子情報通信学会，IEEE 各会員．



勝山光太郎（正会員）

昭和 29 年生．昭和 51 年3月大阪大学基礎工学部制御工学科卒業．同年三菱電機(株)入社．現在同社情報電子研究所システム技術開発部．工学博士．入社以来通信ソフトウェアの開発，分散処理システムの研究開発に従事．本学会研究賞（平成3年度）受賞．電子情報通信学会会員．



水野　忠則（正会員）

昭和 20 年生．昭和 43 年名古屋工業大学経営工学科卒業．同年三菱電機(株)入社．平成5年静岡大学工学部情報知識工学科教授．工学博士．情報通信システムおよび分散処理システムに関する研究に従事．著書としては，「マイコンローカルネットワーク」（産報出版），「MAP/TOP と生産システム」（オーム社），「分散システム入門」（共著，近代科学社），「分散システム―コンセプトとデザイン」（共訳，電気書院）などがある．電子情報通信学会，オフィスオートメーション学会，日本経営工学会，IEEE 各会員．

曽我　正和

昭和 11 年2月生．昭和 33 年京都大学電子工学科卒業．昭和 35 年同修士課程卒業．同年三菱電機(株)入社．国鉄郡山操車場自動化システム，リアルタイムコンピュータ MELCOM 350，汎用コンピュータ MELCOM-COSMO，等の CPU およびハードウェアシステムの開発主任を務めた．その後も主に各種開発プロジェクトに従事．平成2年 12 月より情報電子研究所所長．電子情報通信学会会員．