プロトコル検証システムの設計と実現方式

5T-1

山本博章，白鳥則郎，野口正一
東北大学電気通信研究所

1．まえがき

従来，多くのプロトコルは有限状態機械でモデル化され開発されて来た．このようなプロトコルの開発における検証は，仕様記述言語で書かれた仕様をユーザが自分で検証用表現になおし，そして検証アルゴリズムにかけていた．また，検証結果も検証用の表現で与えられるため，ユーザにとって非常に理解性に乏しいものになっていた．これらの問題を解決し，プロトコル開発の生産性を向上させるために，我々は先に，仕様記述言語ＮＥＳＤＥＬ上でプロトコルの検証ができるシステムを提案した．今回は，この検証システムの各構成要素のより具体的な実現方式について報告する．

2．ＮＥＳＤＥＬとＥＸＰＡ

本検証システムでは仕様記述言語としてＮＥＳＤＥＬ，また検証アルゴリズムとしてＥＸＰＡを使う．それらはすでに文献[1,2,3]で与えられているのでそれらを参照されたい．特に，文献[1]では次節の変換を簡単にするためのＥＸＰＡの拡張が述べられている．

3．検証システムの設計

図２に検証システムのブロック図を示す．以下では，ＮＥＳＤＥＬ－ＥＸＰＡ変換とＥＸＰＡ－ＮＥＳＤＥＬ変換について述べる．

3．1．ＮＥＳＤＥＬ－ＥＸＰＡ変換

ＮＥＳＤＥＬ表現とＥＸＰＡ表現（これはＥＸＰＡへの入力である）の間にはいくつかの違いがある．そのために，ＮＥＳＤＥＬで記述されたプロトコルを検証するためには，対応するＥＸＰＡ表現に変換する必要がある．この章では，ＮＥＳＤＥＬ表現をＥＸＰＡ表現に変換するアルゴリズムを与える．パータベーション解析に基ずいた従来の検証法では，その入力として述語も変数も含まない有限状態遷移図を要求した．しかし，前章で見たようにＮＥＳＤＥＬ表現は述語や変数を持つ有限状態遷移図に基ずいている．従って，変換のアルゴリズムは難しくなる．そのためにユーザは自分で変換を行なっていた．この困難を越えるために，我々はＥＸＰＡ表現の中に述語，変数を導入し，そして，その入力として述語や変数を持ったものも受入れることができるように元のＥＸＰＡを拡張した．これによって，変換アルゴリズムを容易に構成することができるようになった．　以下で，我々はアルゴリズムの基本的な考えを，いくつかの変換規則を与えることによって与える．

アルゴリズム

ＮＥＳＤＥＬのグラフ表現，Ｇ，に対し，次の規則を適用する．

規則１：上位層のプロセスとの通信に関するラベルを取る．ただし，枝はそのまま．

規則２：“timer”，“timeout”に関するラベルを持つ枝は除去する．

規則３：ラベル［transmit,message］を［－，integer］に変換する．

規則４：［put,message］を［＋，integer］に変換する．

規則５：述語及びノード内の演算はそのまま．

上のアルゴリズムに関するいくつかの注意を与える．規則３，４で必ず異なるmessageは異なるintegerに対応させる．基本的には深さ優先探索法を使う．すなわち，Ｇの根よりスタートし，逐次Ｇのノード及び枝に上の規則を適用していく．図１のグラフにこのアルゴリズムを適用した例を図３に示す．

A Design of New Protocol Validation System
Hiroaki YAMAMOTO, Norio SHIRATORI, Shoichi NOGUCHI
Research Institute of Electrical Communication, Tohoku University

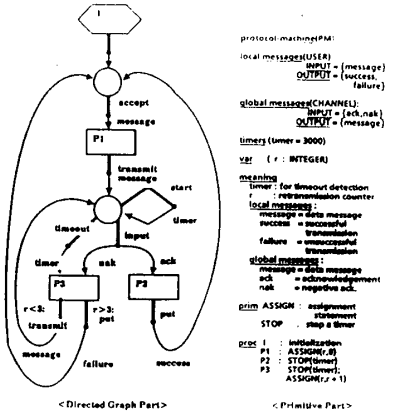

図1. NESDELの記述例

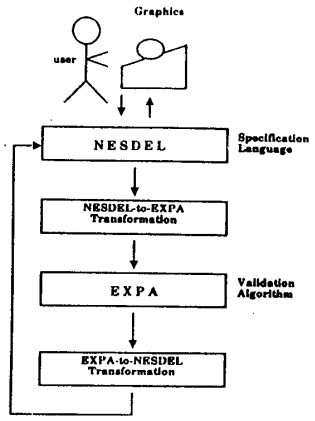

図2 ブロック図

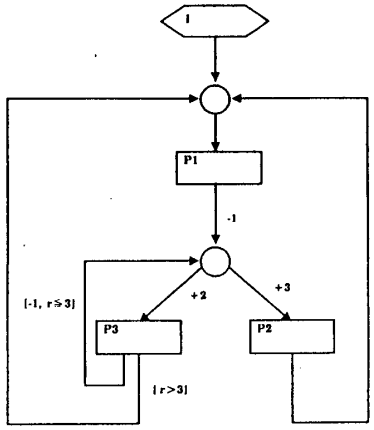

図3. 変換例

3．2 EXPA－NESDEL変換

EXPAはシステム状態の到達可能性グラフを生成し，そして，もし論理エラーが存在すればそれをユーザに知らせる．しかし，ユーザがそれらを理解することは難しい．これは理解性の点においてEXPA表現はNESDEL表現に劣るからだ．そのために我々は，ユーザの理解性を向上させるために，NESDEL表現上で論理エラーを得たい．

以下で，我々はこれを行なうアルゴリズムを与える．このアルゴリズムは2つのステージからなる．1つは分解ステージと呼ばれるもので，これはシステム状態及びシステム状態の列をシステム状態の構成要素である個々のプロセスの状態に分解するものである．もう1つは変換ステージと呼ばれるもので，これは分解ステージの結果をNESDEL表現上に持っていくものである．NESDEL表現は前もって与えられているから，変換ステージの実行は難しくない．アルゴリズムは次の様になる．

アルゴリズム

（I）分解ステージ

Case1 システム状態の分解

Sをシステム状態とする．

（1）SからEXPA表現上で各プロセスの状態を見つけなさい．

（2）NESDEL表現とEXPA表現間の変換情報を使って（1）に対応する状態をNESDEL表現上で見つけなさい．

Case2 システム状態の列の分解

Sをシステム状態の列とし，$S_1 \rightarrow \cdots \rightarrow S_k$とする．ここで，各$S_j$はシステム状態．また，$A_i$は変数でシステム状態の列を値として取るとする．

(1) すべてのiに対し，$A_i \leftarrow q_i$，ここで，q_iはS_1におけるプロセスIの状態．

(2) j＝1からk－1まで以下をくりかえす．

もし$S_j \rightarrow S_{j+1}$がプロセスIの遷移によって引起されたものならば，$A_i \leftarrow A_i * (q', j)$，ここで，$q'$は$S_{j+1}$におけるプロセスIの状態であり，また“＊”は列と列との連接を意味する．

（II）変換ステージ

NESDELからEXPAへの変換情報を使うことにより，このステージは簡単に行なえる．例えば，EXPA表現上での状態遷移が与えられれば，我々は自動的にNESDEL上のどの遷移が対応するかを知ることができる．

4．むすび

我々は先に提案した新しい検証システムのより詳細な実現方式を与えた．さらに，我々はユーザ・インターフェイスをより向上させるため，グラフィック・ディスプレイを使う予定である．現在，EXPA，NESDEL－EXPA変換のインプリメントが終了し，EXPA－NESDEL変換及びグラフィックス関係のインプリメント中である．

参考文献

（1）山本，白鳥，野口，EXPA－IIを用いたプロトコル検証システムの構成，情報処理全国大会，昭和61年前期

（2）白鳥，郷原，野口，EXPA：パータベーション解析に基ずく通信プロトコルの検証法，情報処理，Vol．26，No．3，1985

（3）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，情報処理，Vol．26，No．6，1985