SDL記述のプロトコル検証のためのモデル

2F－1

新田文雄　伊藤篤　宇都宮栄二　斎藤博徳
国際電信電話株式会社　研究所

1．まえがき

信頼性の高い通信システムを設計するためには、信号の送受信規則(プロトコル)の正当性を仕様の段階で検証するプロトコル検証は重要な技術である。しかし、多くのプロトコル検証技術は単純なプロトコルモデル(状態遷移モデル)の上で研究され、SDL[1]の様な拡張有限状態機械(EFSM)に基づく仕様記述言語で記述されたプロトコルには簡単に適用できないという問題点がある。

プロトコル検証における最も基本的なモデル[2]は、プロセス(process)間のチャネル（channel）を通してやりとりされる信号（signal）の送信（send）及び受信（receive）によりプロセスの状態（state）が遷移するFSMに基づく状態遷移モデルである。本稿では、これを基本プロトコルモデルと呼ぶ。多くのプロトコル検証技術は、この基本プロトコルモデルをもとに研究されている[3]。一方、実システムのプロトコルは、SDLなどのEFSMに基づく仕様記述言語を用いて記述されることが多い。そこで、基本プロトコルモデルをもとに研究されたプロトコル検証技術をEFSMに基づく仕様記述言語で記述された仕様に容易に適用可能とするため、新しいプロトコルモデルを提示し、このプロトコルモデルに変換する方法を提案する。本稿では、EFSMに基づく仕様記述言語としてSDLを例にとり、SAVE機能を中心にプロトコルモデルに変換する方法を示す。

2．拡張プロトコルモデル

SDLの基本的な概念(即ちFSM部分)である、PROCESS、STATE、INPUT、OUTPUT、SIGNALは、基本プロトコルモデルのprocess、state、receive、send、signalにそれぞれマッピングできる。一方、SDLはEFSMに基づくため、基本プロトコルモデルにマッピングできない概念がある。例えば、信号保存(SAVE)、条件分岐(DECISION)、条件遷移(PROVIDED)、タスク(TASK)、タイマーなどである。FSMに含まれないこれらのSDL概念は、条件(condition)および演算(operation)という２つの機能を基本プロトコルモデルに導入することにより変換可能となる。このモデル(以下、拡張プロトコルモデルと呼ぶ)は、条件が真の時のみ信号の送信あるいは受信とともに遷移が起こり、その遷移中に演算を行うものである。この拡張プロトコルモデルに基づく言語PDL(Protocol Description Language)を次のように定義する。

<state transition> ::= (<from_state>,<to_state>,<signal_exchange>[,([<condition>],[<operations>])])
<signal_exchange> ::= ([{+|-}<signal_name>,<partner_process>])]
<operations> ::= <operation>{,<operation>}*

ここで、<from_state>と<to_state>は、遷移前の状態と遷移後の状態を示す。プラス記号は、<signal_name>で示される信号の受信を示し、マイナス記号は送信を示す。<partner_process>は、信号を送受信する相手プロセスを示す。

3．SAVE機能の拡張プロトコルモデルへの変換

ここでは、FSMに含まれないSDL概念のうち、SAVE機能の拡張プロトコルモデルへの変換について詳細に述べる。

SAVEされた信号は、後続の連続するSTATEでその信号の受信記述がある限り、SAVEされた信号がなくなるまで受信され続ける。そして、SAVEされた信号があるにもかかわらず、その信号の受信記述がない状態に至った場合には、その信号は廃棄される。この受信記述がない状態は、設計者が意図した場合と誤って

Protocol Model for Protocol Validation of Specifications in SDL
Fumio Nitta, Atsushi Ito, Eiji Utsunomiya and Hironori Saito
KDD R&D Laboratories
2-1-15 Ohara Kamifukuoka-shi, Saitama 356, Japan

書き損なった場合の2種類が考えられる。後者の場合を考慮すると、この受信記述のない状態を未定義受信(プロトコルエラーの一種で、起こり得る信号受信の記述がないことを表す)として検出し、ユーザに通知することが望ましい。この未定義受信の検出を行うためには、プロトコル検証ツールは、次の2つの条件を満足する状態を検出しなければならない。

●SAVEされた信号の数が1以上である状態。

●SAVEされた信号の受信が記述されていない状態。

これを実現するためには、拡張プロトコルモデルの演算機能が利用可能である。具体的には、同一の信号がSAVEされる回数を表すために演算機能が利用される。

以下にSDLで記述した仕様例(左側)と、対応するPDLでの記述(右側)を示す。

```
STATE S1;
        SAVE SIG1;                      (S1,S1,(+SIG1,PARTNER),(,SavedSIG1:=SavedSIG1+1))
        INPUT SIG2;                     (S1,S2,(+SIG2,PARTNER))
                NEXTSTATE S2;           (S1,S3,(+SIG3,PARTNER))
        INPUT SIG3;                     (S2,S4,(),(SavedSIG1>0,SavedSIG1:=SavedSIG1-1))
                NEXTSTATE S3;           (S2,S4,(+SIG1,PARTNER),(SavedSIG1=0,))
STATE S2;                               (S3,S5,(+SIG4,PARTNER),(,SavedSIG1:=0))
        INPUT SIG1;
                NEXTSTATE S4;
STATE S3;
        INPUT SIG4;
                NEXTSTATE S5;
```

右側のPDLでの記述において、SavedSIG1は、SAVEされた信号(SIG1)の数を表すために使われる変数である。1行目の加算は、信号がSAVEされる毎に実行される。4行目は、状態S2において、前状態S1でSAVEされている信号があるかどうかを条件SavedSIG1>0によって検査し、真ならばSAVEされている信号を受信し、偽ならばこの状態遷移は起こらないことを表す。この行の減算は、SAVEされている(1個以上の)信号のうち、最初の一つを受信することを表す。5行目は、状態S2において、前状態S1でSAVEされている信号がなければ、この状態S2で信号SIG1が受信できることを表す。このSAVEされている信号がないことを条件SavedSIG1=0で検査している。6行目では、SAVEされた信号全てを廃棄している。ただし、この状態を未定義受信として検出するために、検証ツールは上記の2つの条件に基づき(1)変数SavedSIG1の値が1以上かどうか検査し、かつ(2)遷移前の状態がS3の場合のSIG1の受信の記述の有無を検査する必要がある。

なお、SAVE機能以外のSDL概念は、基本的には次のような方法で対処できる。条件分岐は、擬似状態を挿入し、拡張プロトコルモデルの条件と演算を利用する。条件遷移とタスクは、それぞれ条件と演算を利用する。タイマーは、条件と演算を利用する。

4. あとがき

本稿では、SDLで記述された仕様のプロトコル検証を行うための拡張プロトコルモデルと、SDLから拡張プロトコルモデルへの変換をSDLのSAVE機能を中心に述べた。

現在、基本プロトコルモデルに基づいたプロトコル検証ツール[4]に対して、拡張プロトコルモデルに対応するようにわずかな改良を加えたプロトコル検証ツール[5]を開発中である。このツールは、入力SDL/PRをPDLに変換した後、検証を行い、検証結果をSDL上に反映させる。

参考文献

[1] CCITT Recommendation Z.100, (1988)

[2] W.C.Lynch: ``Reliable full-duplex transmission over half-duplex telephone lines.", Comm. ACM 11(6), pp.362-372, (1968)

[3] M. T. Liu: ``Protocol engineering", Advance in Computers, 29, pp.79-195, (1989)

[4] 斎藤, 角田: ``プロセス対応状態遷移展開法に基づくプロトコル検証システム", 信学技報 SSE 90-18, pp.31-36, (May 1990)

[5] H.Saito, T.Hasegawa and Y.Kakuda: ``Protocol verification system for SDL specifications based on acyclic expansion algorithm and temporal logic", FORTE '91:Proceedings, pp.513-528, (1991)