マルチメディア通信と分散処理　94－22
分散システム／インターネット運用技術　14－22
（１９９９．７．１６）

タイマシステムコールを用いるFSMプロトコルの適合性試験について

森 亮憲　　樋口 昌宏

大阪大学大学院基礎工学研究科

本論文ではタイマと連係して動作する通信プロトコルの適合性試験について議論する．プロトコル機械は決定性有限状態機械 (DFSM) でモデル化され，システムコールによりオペレーティングシステムのタイマ機能を利用する．タイマの設定状況に着目してタイマを非決定性有限状態機械 (NFSM) でモデル化し，プロトコル機械と合成し試験対象となるシステム全体を NFSM でモデル化する．この NFSM 上で試験系列の候補を従来の手法により生成し，各系列がどのようなタイミングで実行可能になるかを連立不等式を解くことにより求める．得られる系列は DFSM 上の状態遷移における終状態の単一誤り，タイマ操作の単一誤りを検出できる．また，例プロトコルを用いて実際に試験系列が生成できることを確認した．

Conformance Testing of FSM Protocol with Timer Systemcall

Takanori Mori　　Masahiro Higuchi

Graduate School of Engineering Science, Osaka University

In this paper, we discuss the conformance testing of communication protocols which cooperate with timers. Protocol machines are modeled as Deterministic FSM(DFSM) and use timer functions of the operating system through systemcalls. Timers can be modeled as Nondeterministic FSM(NFSM) by classifying states based on timer expireing order. The System Under Test is modeled as NFSM by composing timers and a protocol machine. On this composed machine, we generate test case candidates using a known method for testing NFSM. We determine the executability of candidates by solving simultaneous unequalities. These test cases detect any single faults of destination states or timer operations in the transitions on protocl machines. We also generate test cases for a sample protocol to show the usefulness of our method.

1　まえがき

通信システムの開発において，作成されたシステムがプロトコル仕様通りに動作していることを確認する試験 (適合性試験) は重要である．近年，複数のコンポーネントが互いに連係して動作するような複雑なシステムが使われるようになっている．また，既に稼働しているシステムの一部を利用する形での通信システム開発も行われるようになっている．この場合，動作試験の対象としては，既に稼働しているシステムに関する部分を除外し，新しく開発した部分のみを試験すれば十分である．このようなシステムは既存システム－新規開発システム間のインタフェースが外部から制御観察できない埋め込み型システムとしてモデル化され，主として，既存システムと新規開発システムがともに決定性有限状態機械 (DFSM) とみなせる場合について，試験系列の生成に関する研究が知られている [1][2].

既存システムの一部の機能を利用した通信ソフトウェアの例として，オペレーティングシステムが提供するタイマ機能を利用してタイマ監視を行うシステムを開発する場合が考えられる．この場合，プロトコル機械は DFSM として定義されるとしても，タイマは，通常カウンタを用いて経過時間を測定することから拡張有限状態機械 (EFSM) としてモデル化するのが自然である．しかし，そのようなシステムの試験法として，従来の EFSM に関する試験手法 [5][6] をそのまま適用することはできない．また [1][2] の DFSM モデルを対象とした埋め込み型システムの試験手法も適用できない．本論文ではタイマの設定状況を考慮してタイマを非決定性有限状態機械 (NFSM) でモデル化することを考えた．そして，従来の NFSM 上での試験系列生成手法を用いて新規開発システムに関する単一フォールトを検出するための試験系列の候補を生成する．さらに，EFSM の試験系列生成で用いた手法を使って，生成した試験系列の候補の実行可能性の判

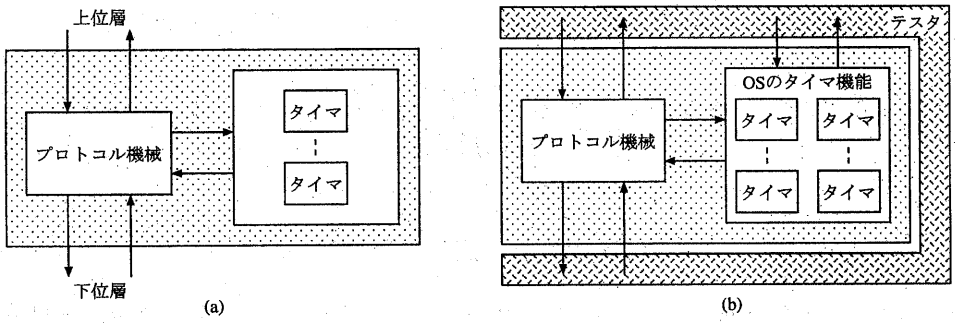


図 1: (a) 試験対象となるシステム　(b) 試験アーキテクチャ

定する.

以降, 2 では本論文で対象としているシステムと試験アーキテクチャについて, 3 では試験対象となるシステムのモデル化について述べる. そして, 試験系列とその実行可能性について 4 で説明し, 5 では例プロトコルを用いた試験系列の生成について述べる.

2 準備

2.1 タイマシステムコールを用いるプロトコル

試験対象として図 1(a) のような一般に複数のタイマを用いる通信プロトコルを考える. タイマはオペレーティングシステム (OS) が提供するタイマ機能を利用すると考える. この場合, プロトコル機械とタイマとのインタフェースは, システムコールと割り込みの形をとる. 通常これらのインタフェースは外部プロセスであるテスタからは観察不可能かつ制御不可能である.

図 1(a) のシステムを試験するために, 図 1(b) のような試験アーキテクチャを考える. このアーキテクチャでは, テスタもシステムコールを用いて時間を計測することができる.

2.2 有限状態機械プロトコル

ここでは, 試験対象となるシステムの各コンポーネントについて説明する.

2.2.1 タイマ

OS のタイマ機能を用いる. OS のタイマ機能は一般に複数のタイマを管理することができ, システムコールによって個々のタイマを設定・解除することができる. 本論文では, 簡単のためにタイマごとに設定できる時間が決まっているものとする. タイマが設定された後, 解除されることなく設定時間が経過すると割り込みによりプロトコル機械にタイムアウトの発生を通知する. また, 設定していないタイマに対して設定を解除する操作を行うと, 戻り値エラーを返す.

2.2.2 プロトコル機械

プロトコル機械は決定性有限状態機械 (DFSM) で定義され, 次のように表現する.

(S, X, T, Y, h, s_0)

S : 状態の有限集合.

X : 外部入力記号の有限集合.

T : タイマの有限集合.

Y : 外部出力記号の有限集合.

h : 状態遷移関数. $(S \times (X \cup T)) \rightarrow (S \times Y \times T_o)$

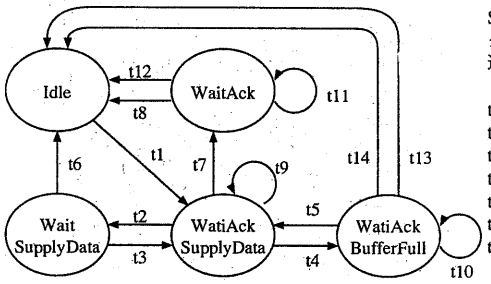
s_0 : 初期状態.

状態遷移は外部から入力が与えられたときまたは, あるタイマがタイムアウトを起こしたときに発生する. 状態遷移関数 h は, 遷移前の状態と入力 (外部入力またはタイムアウトを起こしたタイマ名) から遷移後の状態と出力 (外部への出力とタイマへの出力) を決定する. タイマへの出力は, 各要素が, Settimer, Disable, $-$のいずれかであるような $|T|$ 次元ベクトルである. ベクトルの各要素は個々のタイマへの操作を表す. Settimer はタイマを設定する操作, Disable はタイマ設定を解除する操作を表す. また, $-$は何もしないことを表す.

プロトコル機械はリセット機能を持つ. リセットを行うと, すべてのタイマの設定を解除して初期状態に戻る. また, 定義されていない入力が与えられた場合および設定していないタイマに対して設定を解除する操作を行った場合はエラーとなり, リセットと同様の動作が行われる.

2.3 例プロトコル

以降, 適宜図 2のプロトコルを用いて説明する. これは, ユーザから連続的にデータを受け取り, その



SettimerをS, DisableをDと表す
タイマ1の設定時間 τ 1=7, タイマ2の設定時間 τ 2=25
遷移名：外部入力 / 外部出力, タイマベクトル(1,2)

t1 : SupplyData / SendData, (S, S)
t2 : Ack / Null, (D, -)
t3 : SupplyData / SendData, (S, S)
t4 : SupplyData / Buffer, (-, S)
t5 : Ack / SendData, (S, -)
t6 : Timeout2 / Finish, (-, -)
t7 : Timeout2 / SupplyEnd, (-, -)
t8 : Ack / Finish, (D, -)
t9 : Timeout1 / Resend, (S, -)
t10 : Timeout1 / Resend, (S, -)
t11 : Timeout1 / Resend, (S, -)
t12 : SupplyData / IllegalData, (D, -)
t13 : SupplyData / BufferFULL, (D, D)
t14 : Timeout2 / AckLate(D, -)

図 2: タイムアウト再送を行う連続データ転送プロトコル

データを送出するプロトコルである．ユーザからのデータ供給間隔をタイマ 2 で監視し，一定時間データが供給されないと，ユーザによるデータ転送は終了したと判断する．また，送出したデータに対する Ack を受け取るまで，タイマ 1 の監視によるタイムアウト再送を行う．

3 非決定性有限状態機械によるモデル化

試験系列を生成するためにプロトコル機械とタイマからなるシステムをモデル化する．プロトコル機械は DFSM として与えられる．一方，タイマは通常カウンタを用いて経過時間を測定するので拡張有限状態機械 (EFSM) としてモデル化するのが適当であると考えられる．しかし，タイマには時間を刻む動作があり，この動作は内部的に行われ，タイムアウト発生時以外は外部から観察できない．EFSM プロトコルの試験系列生成についていくつかの研究が知られている [5][6] が，いずれも内部遷移を持つプロトコルをうまく扱うのは困難と考えられる．以下では，内部遷移を持たないモデル化を考える．

3.1 タイママシン

n 個のタイマからなる n タイママシン TM_n のモデル化を考える．このため，タイマの設定状況 (設定されているタイマおよびそのタイマのタイムアウト順序) を考慮して状態を分類することにする．一般性を失うことなくタイマには $1 \sim n$ の名前がついており，タイマ i のタイムアウト時間 τ_i は，$\tau_1 < \tau_2 < \cdots < \tau_n$ となっているものとする．

タイマを n 個用いる場合，設定状況は $\sum_{i=0}^{n} {}_nP_i$ 個になり，TM_n を次のように表すことができる．

$TM_n = (S_T, T_o, h_t, s_\phi)$
S_T : 状態の有限集合．
T_o : 入力ベクトル (n 次元ベクトル) の有限集合．
h_t : 状態遷移関数．$(S \times (T_o \cup \{\lambda\})) \rightarrow 2^S$
s_ϕ : 初期状態．

状態 $[t_1 t_2 \cdots t_k] \in S_T$ は，この後タイマの設定および解除がなければ $t_1, t_2, \cdots, t_k$ の順にタイムアウトが発生する状態を表す．初期状態は，すべてのタイマが設定されていない状態 s_ϕ である．

状態遷移には，タイママシンにタイマ操作ベクトルが入力されて行われる状態遷移と，タイムアウトが発生することで行われる状態遷移の 2 種類がある．

- タイマ操作ベクトルの入力による状態遷移
  タイマ操作ベクトルによりタイマの設定状況が変化する．また，新たに設定されるタイマのタイムアウト順序は決定できない．このため，タイマ操作ベクトルが入力されることによって行われる状態遷移は非決定的な状態遷移となる．
  状態 $[t_1 t_2 \cdots t_k]$ およびタイマ操作ベクトル $v \in T_o$ に対して，状態遷移関数は次のようにする．

$$h_t([t_1 t_2 \cdots t_k],\ v) = \{[x] \mid x \in \delta_v(t_1 t_2 \cdots t_k) || \sigma_v\}$$

  $\delta_v(t_1 t_2 \cdots t_k)$ は，v により設定が解除または再設定されるタイマを $t_1 t_2 \cdots t_k$ から削除する関数である．σ_v は，v によって設定されるタイマの名前を昇順に並べたものを得る関数である．$||$ は，2 つの文字列のインターリーブを表す．
  TM_3 を考える．TM_3 が状態 [23] にいるときに，タイマ操作ベクトル (Settimer, −, Disable) が入力されたとする．状態遷移後の状態は，[12], [21] のいずれかとなる．

- タイムアウトによる状態遷移

状態 s_ϕ を除いて，各状態では必ずタイムアウトが起こる．各状態 $[t_1t_2\cdots t_k]\in S_T-\{s_\phi\}$ に対して，次のように定める．

$$h_t([t_1t_2\cdots t_k],\ \lambda)=[t_2\cdots t_k]$$

タイムアウトするタイマ以外の設定状況は変化しないので，各状態に対してタイムアウトによる状態遷移が一意に定まる．

例えば，TM_3 が状態 [231] にいる場合にタイムアウトが発生すると，状態 [31] に遷移する．

このように，非決定性有限状態機械 (NFSM) ではあるが，内部遷移を持たないようにタイマをモデル化することができる．

タイママシンは NFSM であるが，タイマ操作ベクトルを与えるタイミングを制御することにより決定的な動作をさせることができる．タイママシンを実際に動作させる場合，新たに設定するタイマのタイムアウト順序は，すでに設定されているタイマを設定してからの経過時間によって決まる．つまり，タイマ操作ベクトルを与えるタイミングを制御すればタイマのタイムアウト順序を制御でき，タイママシンに決定的な動作をさせることができる．

3.2 プロトコル機械とタイマの合成

プロトコル機械 $A=(S,X,T,Y,h,s_0)$ とタイママシン $TM_{|T|}=(S_T,T_o,h_t,s_\phi)$ を合成して，$B=\big((S\times S_T),(X\cup\{\mathrm{WE}\}),Y,h',\langle s_0,s_\phi\rangle\big)$ を生成する．

状態遷移関数 h'は次のように定める．$h(s_i,\ x)=(s_j,\ y,\ v)\quad(s_i,s_j\in S,\ x\in X,\ y\in Y,\ v\in T_o)$ であれば，各 $[t_1t_2\cdots t_k]\in S_T$について次のようにする．

$$h'\big(\langle s_i,[t_1t_2\cdots t_k]\rangle,\ x\big)$$
$$=\ \big\{\langle s_j,s'_t\rangle,y\mid s'_t\in h_t([t_1t_2\cdots t_k],v)\big\}$$

また，$h(s_i,\ t)=(s_j,\ y,\ v)\quad(t\in T)$ であれば，$TM_{|T|}$においてタイマ t のタイムアウトによって起こる各状態遷移 $h_t([tt_2\cdots t_k],\ \lambda)=[t_2\cdots t_k]$ に対して次のようにする．

$$h'\big(\langle s_i,[tt_2\cdots t_k]\rangle,\ \mathrm{WE}\big)$$
$$=\ \big\{\langle s_j,s'_t\rangle,y\mid s'_t\in h_t([t_2\cdots t_k],v)\big\}$$

WE はタイマ t のタイムアウトを待つことを表す．

さらに，各状態 $s_i\in S$に対して，次のようにする．

$$h'\big(\langle s_i,s_\phi\rangle,\ \mathrm{WE}\big)=\big(\langle s_i,s_\phi\rangle,\ \varepsilon\big)$$

これはいくら待ってもタイムアウトが起こらないことを表す．実際の試験の際には，これらの状態遷移はテスタがτ_nを計測してその間外部出力がないことによって観察する．

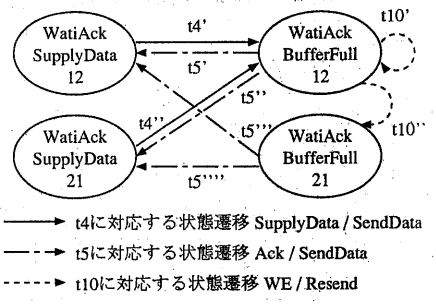


図 3: 例プロトコルと TM_2 を合成した NFSM(一部)

B上では，プロトコル機械とタイママシンのインタフェースは見えなくなるので，外部から Bに対して行えるのは，外部入力を与えることと，入力を与えずに待つことである．よって，上で述べた 3 種類の状態遷移で B上のすべての状態遷移を構成することができる．

例として図 2と TM_2を合成する．合成した機械のうち $\langle WASD,12\rangle$，$\langle WASD,21\rangle$，$\langle WABF,12\rangle$，$\langle WABF,12\rangle$ の 4 状態と状態遷移 t_4, t_5, t_{10}に対応する状態遷移だけを図 3に示す．

プロトコル機械が最簡形であってもプロトコル機械とタイママシンを合成して得られる機械は，最簡形であるとは限らない．プロトコル機械に 2 個の状態 s_1,s_2があり，(1) 各外部入力に対する外部出力と状態遷移先が同じ．(2) それぞれの状態でタイムアウト動作が定義されているタイマの集合 T_1,T_2に，$T_1\subseteq T_2$ または $T_1\supseteq T_2$の関係があり，タイムアウトによる動作が同じ．以上のような 2 つの条件を満たすとき，$\langle s_1,s_t\rangle$ と $\langle s_2,s_t\rangle$ (s_tで T_2-T_1中のタイマは動作していない) は等価な状態になる．

4 試験系列

4.1 フォールトモデル

プロトコル機械の仕様を Aとする．Aにおける各状態遷移について，次のようなフォールトモデルを設定する．(1) 終状態の単一誤り．(2) タイマへの出力の単一誤り．(3) 外部出力の単一誤り．

Aとタイママシン $TM_{|T|}$を合成した B上で，これらのフォールトモデルを考える．A のある状態遷移は，Bにおいて複数の状態遷移に対応する．A の状態遷移に誤りがあると，その状態遷移に対応する B上の状態遷移はすべて誤りとなる．

具体的に各フォールトは次のようになる．Aにおいて終状態が s であるような状態遷移 t を考える．この

状態遷移に対応するB上の状態遷移の集合をT_tとし，それらの遷移の終状態の集合が$\{s\} \times \mathcal{S}$ $(\mathcal{S} \subseteq S_T)$であるとする．(1)の場合，$A$における終状態が誤っているので，$T_t$の各遷移の終状態は$\{s'\} \times \mathcal{S}$ $(s' \neq s)$となる．(2)の場合，タイマへの出力が誤っているのでT_tの各遷移の終状態は$\{s\} \times \mathcal{S}'$ $(\mathcal{S}' \subseteq \mathcal{S},\ \mathcal{S}' \cap \mathcal{S} = \phi)$となる．(3)の誤りは，$t$に対応する$B$上でのすべての状態遷移の外部出力が誤ることになる．

4.2 単一誤りを検出する試験系列

最初に，Aにおける状態遷移の終状態の正しさを試験する試験系列について議論する．Bの初期状態から状態遷移関数を用いて到達可能な状態の集合$S_R \subseteq S \times S_T$を考える．$S_R$の各要素$\langle s, s_t \rangle$について，次のような系列$EIO^{s}_{\langle s,s_t \rangle}$を生成する．

- $EIO^{s}_{\langle s,s_t \rangle}$：$B$が状態$\langle s, s_t \rangle$にいる場合に実行可能であり，状態$\langle s', s_t \rangle$ $(s' \in S,\ s' \neq s)$にいる場合は実行不可能な系列．

系列$EIO^{s}_{\langle s,s_t \rangle}$は，UIO系列の考え方に基づいている．UIO系列は，仕様として与えられる状態集合から特定の状態を同定する系列である．一方，系列$EIO^{s}_{\langle s,s_t \rangle}$は，仕様として与えられる状態集合の部分集合から特定の状態を同定する系列である．系列$EIO^{s}_{\langle s,s_t \rangle}$を生成するアルゴリズムは，UIO系列を生成するアルゴリズム[4]を利用することによって実現できる．

3.2で述べたように，Bには等価な状態が存在する場合がある．等価な状態$\langle s_i, s_t \rangle$, $\langle s_j, s_t \rangle$が存在したとする．このとき，$EIO^{s}_{\langle s_i,s_t \rangle}$は存在しない．そこで，等価な状態は区別する必要がないと考えて，$EIO^{s}_{\langle \{s_i,s_j\},s_t \rangle}$を求める．

A上の状態遷移tの終状態の正しさを調べる試験系列は，次のようにして得られる．B上でのtに対応する状態遷移のうち初期状態から到達可能な状態遷移のみを取り出す．4.1で述べたように，tに対応するB上の状態遷移は“すべて正しい”か“すべて誤っている”のどちらかである．したがって，Bにおける複数の状態遷移の中から一つを選択すればよい．選択した状態遷移の終状態はS_Rに含まれているので，対応する系列$EIO^{s}_{\langle s,s_t \rangle}$がある．状態遷移に対応する入出力の後に，状態遷移の終状態が正しいことを確認するための$EIO^{s}_{\langle s,s_t \rangle}$を連接する．

次に，Aにおける状態遷移のタイマへの出力の正しさを試験する試験系列について考える．Bの初期状態から到達可能な状態の集合S_Rの各要素$\langle s, s_t \rangle$について，次のような系列$EIO^{t}_{\langle s,s_t \rangle}$を生成する．

- $EIO^{t}_{\langle s,s_t \rangle}$：$B$が状態$\langle s, s_t \rangle$にいる場合に実行可能であり，状態$\langle s, s'_t \rangle$ $(s'_t \in S_T,\ s'_t \neq s_t)$にいる場合は実行不可能な系列．

A上の状態遷移tのタイマへの出力の正しさを調べる試験系列は，先程と同様に，B上のtに対応する状態遷移のうち到達可能な状態遷移の中から一つを選び，$EIO^{t}_{\langle s,s_t \rangle}$を連接すれば得ることができる．

外部出力の誤りは，状態遷移を実行したときの外部出力を観察すればよいので，先に述べた2つの試験系列によって試験することができる．

上記の方法で生成した試験系列が実行可能とは限らない．そこで，試験系列が実行可能かどうかを調べる必要がある．試験系列IOは，状態遷移$t_1, t_2, \cdots, t_n$を実行する入出力系列であるとする．状態遷移t_iは状態s_{i-1}からs_iへの遷移であるとする．また，状態遷移t_iを実行する時刻をd_iとする．s_0へ状態遷移した時刻を0とする．状態s_iでのタイマaのタイムアウトまでの時間は，状態遷移t_i終了直後におけるタイムアウトまでの時間で$r(s_i, a)$と表す．状態遷移t_iがタイマaのタイムアウトにより起こる場合は，$d_i = d_{i-1} + r(s_{i-1}, a)$とする．

$s_0, s_1, \cdots, s_n$の各状態について，設定されているタイマのタイムアウトまでの時間を求める．$r(s_j, a)$は，次の式のいずれかで表すことができる．

$$\tau_a - (d_j - d_i) \tag{1}$$

$$r(s_0, a) - d_j \qquad \left(r(s_0, a) \leq \tau_a\right) \tag{2}$$

式(1)は，タイマaが状態遷移t_iで設定された場合，式(2)は，タイマaが状態s_0で設定されていた場合である．さらに各状態におけるタイマの設定状況を満たすように不等式を立てる．例えば状態$\langle s_i, [21] \rangle$については$r(s_i, 2) < r(s_i, 1)$, $d_{i+1} - d_i < r(s_i, 2)$となる．2つ目の不等式を立てるのは，タイマ2が入力を遅らせている間にタイムアウトすることを防ぐためである．これらの不等式からなる連立不等式が非負の解を持てばIOは実行可能である．各不等式は，

$$d_x - d_y \leq c \qquad (1 \leq x, y \leq n,\ c: \text{定数})$$

の形なるので，Bellman-Fordのアルゴリズムを用いて$O(lm)$ (lは不等式の数，mは変数の数)で解けることが知られている[7]．

また，連立不等式を解くことでs_0における各タイマのタイムアウトまでの時間に関する条件も求められる．試験スイートを組む際は，この条件を満たすように先行系列を決定する．

表 1: 初期状態から到達できる状態と各状態に対する EIO 系列

状態	$EIO^{s}_{\langle s,s_t\rangle}$	$EIO^{t}_{\langle s,s_t\rangle}$
⟨Idle, -⟩	SupplyData/SendData	WE/ϵ
⟨WaitAckSupplyData, 12⟩	Ack/Null	WE/Resend, WE/SupplyEnd
⟨WaitAckSupplyData, 21⟩	Ack/Null	WE/SupplyEnd, Ack/Finish
⟨WaitSupplyData, 2⟩	WE/Finish	WE/Finish, WE/ϵ
⟨WaitAckBufferFull, 12⟩	Ack/SendData	WE/Resend, WE/AckLate
⟨WaitAckBufferFull, 21⟩	Ack/SendData	WE/AckLate
⟨WaitAck, 1⟩	Ack/Finish	Ack/Finish, WE/ϵ

図 3では例プロトコル上の状態遷移 t_5に対応する状態遷移が 4 つある．$t5''$の終状態は ⟨WASD, 21⟩ であり，$EIO^{s}_{\langle \mathrm{WASD},21\rangle}$ = Ack/Null である．試験系列 Ack/SendData,Ack/Null が実行可能かどうかを調べる．連立不等式は次のようになる．

$$
\begin{array}{ll}
d_2 - d_1 \geq 0, & r(s_0,1) < r(s_0,2), \\
0 < r(s_0,1) - d_1, & r(s_0,2) - d_1 < \tau_1, \\
0 < r(s_0,2) - d_2 & \text{ただし } s_0 = \langle \mathrm{WABF}, [12] \rangle
\end{array}
$$

d_1, d_2は解を持つので，試験系列は実行可能である．また，s_0でのタイマに関する条件は$r(s_0,2)$–$r(s_0,1) < \tau_1$である．

5 例プロトコルへの適用

図 2の例プロトコルに対して，手作業により試験系列を生成した．このプロトコルとTM_2を合成すると，状態数 25 の NFSM ができる．初期状態から到達できる状態は 7 個で各状態に対する EIO 系列は表 1のようになった．

t_5の終状態の正しさを調べる試験系列を考える．t_5の終状態は WaitAckSupplyData である．4.2の例で用いた試験系列を使う．先行系列としてt_1, t_4, t_{10}, t_{10}を選択する．入力を与える場合は間隔を取らないとすると，$r(s_0,1) = 7, r(s_0,2) = 11$ となり，s_0での条件を満たす．さらに，$4 < d_1 < 7$, $d_1 \leq d_2 < 7$ が得られる．$d_1 = 5$, $d_2 = 5$ とすると，初期状態からの入出力系列は以下のようになる．

SupplyData/SendData, SupplyData/Buffer,
WE/Resend, WE/Resend, Settimer(5)/–,
WE/Timeout, Ack/SendData, Ack/Null

Settimer(5) は，テスタが 5 単位時間を計測するために，タイマに与える入力である．また Timeout は，Settimer(5) に対するタイムアウトである．

6 まとめ

本論文では，タイマを用いる通信プロトコルの適合性試験について議論し，各遷移に対する終状態の正しさ，タイマ操作の正しさを試験する試験系列を提案した．タイマは信頼できるコンポーネントであると考え，試験対象となるシステム全体を NFSM でモデル化した．また，テスタもタイマ機能を利用するような試験アーキテクチャを考えた．提案した試験系列は UIO 系列の考え方に基づいており，状態を同定することによって単一誤りを検出する．例プロトコルに対して手作業で試験系列の生成を試み，試験系列が生成できることを確認した．

参考文献

[1] A.Petrenko, N.Yevtushenko: Fault detection in embedded components, Testing of Communicating Systems, pp.272-287(1997)

[2] L.P.Lima Jr, A.R.Cavalli: A Pragmatic Approach to Generating Test Sequences for Embedded Systems, Testing of Communicating Systems, pp.288-307(1997)

[3] A.Petrenko, N.Yevtushenko, A.Lebedev, A.Das: Nondeterministic State Machine in Protocol Conformance Testing, Protocol Test Systems, VI, pp. 363-378(1994)

[4] K.Sabnani, A.Dahbura: A Protocol Test Generation Procedure, Computer Networks and ISDN Systems, vol.15, no.4, pp.285-297(1988)

[5] C.J.Wang, M.T.Liu: Axiomatic Test Sequence Generation for Extended Finite State Machines, Proc 12th Int'l Conf. on Distributed Computing Systems, pp. 252-259(1992)

[6] 樋口昌宏, 小原勝, 中石敬治, 藤井護: 通信プロトコル適合性試験におけるレジスタ操作に対する試験系列の生成手法, 情報処理学会論文誌, vol.39, no.4, pp.1067-1076(1998)

[7] T.H.Cormen, R.L.Rivest: Introduction to Algorithms, The MIT Press, pp.539-543(1990)