

あるクラスの時間オートマトンに対する 適合性試験系列生成の一手法

深 田 敦 史[†] 中 田 明 夫^{††}
東 野 輝 夫[†] 谷 口 健 一[†]

本論文では、過去の動作の実行時刻を時間制約に使用できるようにある時間オートマトンモデルを提案し、そのモデル上での1つの適合性試験法を提案する。時間オートマトンモデル上の試験において、与えられた試験系列(I/O動作系列)を実行可能にするためには、その系列中のすべての動作(遷移)の時間制約を満たすようなタイミングで入出力動作を実行しなければならない。しかし、出力動作はシステムからの反応であるため、そのタイミングをあらかじめ指定することはできない。また、そのタイミングに依存して以降の入出力動作の実行タイミングも変化する。そのため、一般に試験系列中の各入力動作の実行可能時刻はその入力動作に先行する動作が実行された時刻の関数で表現される。本論文では、線形計画法の手法を用いて、各試験系列の実行可能性の判定と実行可能な場合の各入力動作の実行可能時刻を表す関数を機械的に求める方法を与え、それに基づいた適合性試験法を提案する。

Generation of Test Cases with Timing Constraints from Timed I/O Automata

ATSUSHI FUKADA,[†] AKIO NAKATA,^{††} TERUO HIGASHINO[†]
and KENICHI TANIGUCHI[†]

In this paper, we propose a timed I/O automaton model and a conformance testing method for the model. In order to trace a test sequence (I/O event sequence) on the model, we need to execute each I/O event at an adequate execution timing which satisfies all timing constraints in the test sequence. However, since the outputs are given from the IUT and uncontrollable, we cannot designate their output timing in advance. Also each output timing may affect the executable timing for the succeeding I/O events in the test sequence. Therefore, in general, the executable timing of each input event in a test sequence can be specified by a function of the execution times of the preceding events. In this paper, we propose an algorithm to decide whether a given test sequence is executable. Then, we give an algorithm to derive such a function from an executable test sequence using a technique for solving linear programming problems, and propose a conformance testing method using the algorithms.

1. ま え が き

通信ソフトウェアの信頼性を高める1つの手法として適合性試験があり、多くの研究がなされている⁷⁾。しかし、時間を考慮したモデルに対する試験法については十分に研究されているとはいえない。しかし近年、時間オートマトンモデル¹⁾やいくつかの実時間プロト

コルに対する試験法が提案されてきている^{3),9)}。

一般にEFSMや時間オートマトンモデルでの試験では、オートマトン上で1つの試験系列(遷移系列)を与えても、その遷移系列が必ず実行できるとは限らず、実行可能な場合も、その系列中の各遷移の遷移条件を満足するような適切な入力データや実行タイミングを指定する必要がある。EFSMモデルでは、ヒューリスティックを用いたり、遷移条件のクラスを制限したりすることにより、適切な入力データ値を機械的に求める手法がいくつか提案されている^{2),4),12)}。しかし、実時間プロトコルの試験では、テスターからIUT(Implementation Under Test)への入力は指定したタイミングで与えることができて、テスターへの出

[†] 大阪大学基礎工学部情報科学科

Department of Information and Computer Sciences,
Faculty of Engineering Science, Osaka University

^{††} 広島市立大学情報科学部情報数理学科

Department of Computer Science, Faculty of Information Sciences, Hiroshima City University

力は IUT の出力タイミングを制御できないので、そのタイミングをあらかじめ固定できない。また、その実行タイミングに依存して、後続の入出力動作の実行可能時刻が変化する可能性がある。このため、試験に際しては、その系列に含まれる出力動作が指定された範囲内のどの時点で行われても、与えられた試験系列を実行可能とするような入力動作の実行タイミングが存在することが望ましい。そこで、本論文では実時間プロトコルの仕様を記述するための I/O 時間オートマトンモデルを提案し、そのモデル上で上述の点を考慮した 1 つの適合性試験手法を提案する。

提案する手法では、各遷移が入力か出力のいずれかであるようなあるクラスの I/O 時間オートマトンで仕様を記述する。そのモデルでは、システム全体の現在時刻（実数上の時間）を表すグローバルなクロック変数と過去の動作の実行時刻などを記憶するための変数を導入し、各遷移の実行条件として、それらの変数の線形不等式の論理積からなる任意の論理式が記述できる。それらの変数には、各遷移の実行時に他の変数と入力値を表す変数からなる一次式を代入できる。このモデルを用いて、2 つの動作の実行時刻の差に依存した遷移条件（たとえば、引き続き 2 つの受信動作間の時間差がある閾値を超えたら別の割り込み動作を行うこと）を記述したり、その時間差だけ待って別の動作を行うなど、QoS 制御を行うマルチメディア通信プロトコルなどに典型的に現れる処理を簡潔に記述できる。

与えられた試験系列の実行可能性としては、どんな出力タイミングに対しても後続の入力動作の実行タイミングが存在する場合（must トレース可能性）と、適当な出力タイミングに対して（のみ）後続の入力動作の実行タイミングが存在する場合（may トレース可能性）、の 2 通りがある。本論文では、各遷移の実行条件を線形不等式の論理積に制限することにより、must/may トレース可能性を効率良く機械的に判定する方法を提案する。また、その試験系列に含まれる各入力動作の実行可能時間幅を先行する動作の実行時刻の関数として与える方法を提案する。

さらに、これらのアルゴリズムを用いて UIOv-法¹¹⁾に基づく 1 つの適合性試験法を考案した。提案する試験手法では、IUT が仕様の状態数を超えない数の状態数からなる I/O 時間オートマトンで与えられると仮定できる場合、(a) 仕様の各状態に対応する状態が IUT にも存在すること、(b) 仕様と同じ遷移が IUT にもあること（各状態からの遷移に過不足がなく、かつ、各遷移の開始状態や遷移先状態に誤りがないこと）を保

証できる。また、遷移条件の実装の正しさについては、EFSM の試験の場合^{2),7)}と同様、実装の誤りをいくつかの場合に特定し、特定した誤りがあるかどうかを試験する方法を提案する。

以下、2 章で提案する I/O 時間オートマトンを定義し、3 章ではそのモデル上での試験系列の実行可能性について述べる。4 章で UIOv-法を応用した適合性試験の一手法を提案し、5 章でその適用例を示す。6 章でまとめを述べる。

2. I/O 時間オートマトン

2.1 I/O 時間オートマトンの定義

定義 2.1 $M = \langle S, A, I/Otype, t, V, Pred, Def, \delta, s_{init}, \{x_{1init}, x_{2init}, \dots, x_{kinit}\} \rangle$ を I/O 時間オートマトンと定義する。ここで、

- S は M の状態の有限集合、 $S = \{s_0, s_1, \dots, s_n\}$
- A は M の入出力動作名の有限集合
- $I/Otype = \{!, ?\} \cup \{?_v | v \in V\}$. 記号 $!$, $?$ はそれぞれ入力、出力を表す。入力動作において、入力値 v を変数に代入し以降の遷移条件などに利用したい場合は $?_v$ のように表す。出力値については、通常直接以降の遷移条件に影響することがないので、このモデルでは省略する。
- 現在時刻を表すグローバルなクロック変数 t
- V は変数の有限集合、 $V = \{x_1, x_2, \dots, x_k\}$
- $Pred$ は線形不等式 $P[t, x_1, x_2, \dots, x_k]$ の論理積からなる論理式の集合
- Def は一次式 $f(t, v, x_1, x_2, \dots, x_k)$ の変数 $x_i \in V$ への代入を表す代入文 $x_i \leftarrow f(t, v, x_1, x_2, \dots, x_k)$ の集合
- δ は遷移関数。 $S \times A \times I/Otype \times V \rightarrow S \times V$
- $s_{init} \in S$ は M の初期状態
- $\{x_{1init}, x_{2init}, \dots, x_{kinit}\}$ はそれぞれ変数 $x_1, x_2, \dots, x_k \in V$ の初期値

である。また、モデル M 上の遷移は、 $s, s' \in S, a \in A, \$ \in I/Otype, P \in Pred, D \subseteq Def$ のとき、 $s \xrightarrow{a\$[P]D} s'$ と表記する。

I/O 時間オートマトンの遷移の動きの直観的な意味は次のとおりである。たとえば、状態 s での変数 $x_1, x_2 \in V$ の値がそれぞれ 2.5, 4.7 であるとする。2 つの遷移 $s \xrightarrow{a?[x_1 \leq t \wedge t \leq 5.2 + x_2 - x_1]\{x_1 \leftarrow t\}} s' \xrightarrow{b![t \leq x_1 + 2]\{x_1 \leftarrow t + 3\}} s''$ は次のような意味を持つ。まず状態 s で、 $2.5 \leq t \leq 5.2 + 4.7 - 2.5$ を満足するような時刻 t 、つまり、時刻 2.5 以降 7.4 (= 5.2 + 4.7 - 2.5) 以内ならば入力動作 $a?$ が実行可能であり、実行可能時刻に入力 a を実行すれば、状態 s' に遷移し、代入

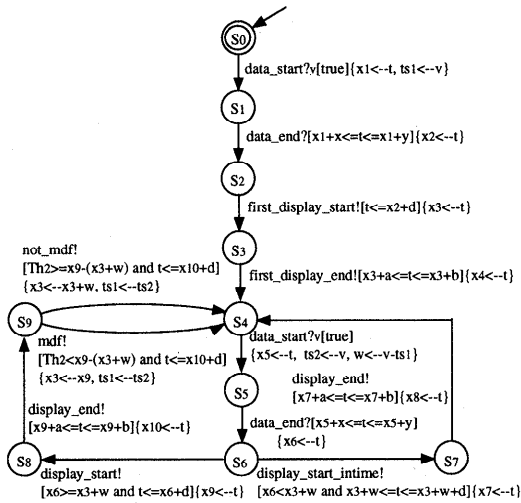


図1 メディア同期制御プロトコル

Fig. 1 Media synchronization protocol.

文 $\{x_1 \leftarrow t\}$ に基づき実際に入力動作 $a?$ が実行された時刻 t が変数 x_1 に代入される。次に、この $a?$ の実行時刻が5であったとすると、変数 x_1 の値は5であるので、出力動作 $b!$ は、遷移条件 $t \leq 5 + 2$ を満足するような時刻 t 、つまり、時刻7以内に実行されることを表す。 $b!$ の実行時刻が6であると、変数 x_1 に $t + 3$ 、つまり $6 + 3 = 9$ を代入し、遷移先の状態 s'' に移る。

なお、 $a?v$ を用いた遷移 $s \xrightarrow{a?v[t \leq 10]\{x_1 \leftarrow v\}} s'$ では、入力動作 $a?$ の実行により、外部から入力値を受け取ることができる。その遷移の代入文中でその入力値 v を変数 x_1 に代入するように記述することにより、以降その入力値を遷移条件などに使用できる。

2.2 I/O 時間オートマトンにおける記述例

図1は、蓄積された連続メディアの転送を行う際の受信側のメディア内同期制御（急激な同期誤差回復）⁵⁾ を行うプロトコルを一部修正した例である。

このシステムでは、データ送信側は受信側に向けてある間隔でデータを連続して送信する。まず送信側は、送信時刻のタイムスタンプを付けたデータを受信側に向けて送信する。受信側はそのデータを時刻 x_1 で受信し始め、同時にそのタイムスタンプの値 (ts_1) を読み取る。そして、ネットワークの遅延時間（最小遅延時間 x 、最大遅延時間 y ）経過後のある時刻 x_2 にその受信を完了する。次に、受信側のシステムが出力を行う準備が整うまでにかかる最大時間 d までに、そのデータの画面表示を開始する信号を出力する。MPEGデータのような場合、データのデコードに要する時間を考慮し、データの表示完了までにかかる時間（最小

時間 a 、最大時間 b ）後に、データの画面表示を終了する信号を出力する。それ以降も、受信側は送信側からデータを受信しその表示を行うが、その際データ再生間隔の同期を行う。具体的には、前回のデータの画面表示開始時刻 (x_3) から送信側における前回のデータと今回のデータの送信時間間隔だけ経過した時刻、つまりそれら2つのデータのタイムスタンプの値の差 (w) と同じ時間経過した時刻 ($x_3 + w$) を、次の受信データの目標画面表示開始時刻にする。もしも、実際のデータ受信完了時刻が目標の画面表示開始時刻よりも早い時刻 ($x_6 < x_3 + w$) であれば、目標時刻になるまで待ってから表示開始の信号を出力する。一方、データ受信完了時刻が目標時刻以降の場合は、その時点（時刻 x_9 ）ですぐに表示開始の信号を出力する（急激な同期誤差回復）。また、このように目標時刻よりもデータの表示開始が遅れた場合でも、画面表示開始時刻が目標時刻に比べある閾値 (Th_2) 以内 ($Th_2 \geq x_9 - (x_3 + w)$) であれば、このデータの画面表示は目標時刻どおりに行われたとする ($not_mdf!$)。しかし閾値を超えた場合 ($Th_2 \leq x_9 - (x_3 + w)$) は、そのデータの実際の画面表示開始時刻 x_9 を x_3 に代入し、同期時間間隔を修正する ($mdf!$)。以降、同様にデータの受信、表示を繰り返す。なお、図1中の a 、 b 、 d 、 x 、 y 、 Th_2 は、試験実施時に定数として与えられるものとする。

3. 提案するモデル上での系列の実行可能性

3.1 must/may トレース可能性

I/O 時間オートマトンでは、各遷移の実行時に変数の値を更新していくため、ある遷移系列の実行可能性を判定するためには、各遷移の実行で変数の値がどのように変化していくかを考慮する必要がある。そこで、与えられた遷移系列中の各変数の値や遷移条件を、変数の初期値や遷移の実行時刻、入力値を表す変数からなる式に置き換える。

定義 3.1 仕様 M 上の初期状態 $s_0 = s_{init}$ からの遷移系列 $s_0 \xrightarrow{a_1 s_1 [P_1] D_1} s_1 \xrightarrow{a_2 s_2 [P_2] D_2} \dots \xrightarrow{a_n s_n [P_n] D_n} s_n$ （以下この系列を α とする）に対して、 $a_1, \dots, a_n$ の実行時刻を表す変数 $t_1, \dots, t_n$ と、系列 α 中にある m 回のデータ入力動作により得られた入力値を表す変数 $v_1, \dots, v_m$ を用いて、状態 s_i における変数 $x_1, \dots, x_k \in V$ の値 $x_1^{(i)}, \dots, x_k^{(i)}$ を次のようにして、各々 $t_1, \dots, t_i, x_{linit}, \dots, x_{kinit}, v_1, \dots, v_m$ からなる式で表す。

- $j := 0$, for $p = 1$ to k do $x_p^{(0)} := x_{pinit}$
- for $i = 1$ to n do

$$\begin{aligned}
TrCondMust(w) &\stackrel{\text{def}}{=} \exists v_1 \dots \exists v_m [TrCondMust'(w, 0)] \\
TrCondMust'(\varepsilon, t) &\stackrel{\text{def}}{=} true \\
TrCondMust'((a_i \$i, t_i, P_i)w, t) &\stackrel{\text{def}}{=} \exists t'_i [P_i[t'_i/t_i] \wedge (t \leq t'_i)] \\
&\quad \wedge \forall t_i [(P_i \wedge (t \leq t_i)) \Rightarrow TrCondMust'(w, t_i)] \\
&\quad \text{if } \$i = !, \\
&\quad \exists t_i [P_i \wedge (t \leq t_i) \wedge TrCondMust'(w, t_i)] \\
&\quad \text{otherwise.} \\
TrCondMay(w) &\stackrel{\text{def}}{=} \exists v_1 \dots \exists v_m [TrCondMay'(w, 0)] \\
TrCondMay'(\varepsilon, t) &\stackrel{\text{def}}{=} true \\
TrCondMay'((a_i \$i, t_i, P_i)w, t) &\stackrel{\text{def}}{=} \exists t_i [P_i \wedge (t \leq t_i) \wedge TrCondMay'(w, t_i)]
\end{aligned}$$

図2 must/may トレース可能性

Fig. 2 must/may traceability.

- if $\$i = ?v$ then $j := j + 1$; $\$i := ?v_j$
- for $p = 1$ to k do
 - * if $x_p \leftarrow f(t, v, x_1, \dots, x_k) \in D_i$
 - $x_p^{(i)} := f(t_i, v_j, x_1^{(i-1)}, \dots, x_k^{(i-1)})$
 - * else (x_p への代入文が D_i に存在しない)
 - $x_p^{(i)} := x_p^{(i-1)}$

これにより得られた $x_1^{(i)}, \dots, x_k^{(i)}$ を用いることにより、各遷移条件 $P_i[t, x_1, \dots, x_k]$ は、

$$\begin{aligned}
\bar{P}_i &\stackrel{\text{def}}{=} P_i[t/t, x_{1init}/x_1, \dots, x_{kinit}/x_k] \\
\bar{P}_i &\stackrel{\text{def}}{=} P_i[t_k/t, x_1^{(i-1)}/x_1, \dots, x_k^{(i-1)}/x_k]
\end{aligned}$$

のように表せる。ここで $P_h[t'/t, x'_1/x_1, \dots, x'_k/x_k]$ は $P_h[t, x_1, \dots, x_k]$ の変数 $t, x_1, \dots, x_k$ を、それぞれ、式 $t', x'_1, \dots, x'_k$ に置き換えて得られる式を表す。このとき、 $w \stackrel{\text{def}}{=} (a_1 \$1, t_1, \bar{P}_1) \dots (a_n \$n, t_n, \bar{P}_n)$ を系列 α に対するシンボリックトレースと定義する。

例 3.1 仕様 M 上のループを含むある遷移系列 $s_0 \xrightarrow{a[t \leq x_b + 7]\{x_a \leftarrow t\}} s_1 \xrightarrow{b?v[t \leq x_a + 3]\{x_b \leftarrow t, x_a \leftarrow x_a + v\}} s_2 \xrightarrow{c[t \leq x_b + 5 \wedge x_a + 15 \leq t]\{t\}} s_0 \xrightarrow{a[t \leq x_b + 7]\{x_a \leftarrow t\}} s_1$ に対するシンボリックトレース w は、 $w = (a!, t_a, t_a \leq x_{binit} + 7) (b?v_1, t_b, t_b \leq t_a + 3) (c?, t_c, t_c \leq t_b + 5 \wedge t_a + v_1 + 15 \leq t_c) (a!, t'_a, t'_a \leq t_b + 7)$ である。たとえば、ここで最後の $(a!, t'_a, t'_a \leq t_b + 7)$ の遷移条件 $t'_a \leq t_b + 7$ は、本来の遷移条件 $t \leq x_b + 7$ の時刻 t をこの動作の実行時刻 t'_a で置き換え、変数 x_b を先行する動作 $b?v$ の代入文 $x_b \leftarrow t$ に基づき $b?v$ の実行時刻 t_b で置き換えることによって得られる。

シンボリックトレース w に対して、 w 中の各出力動作がどんなタイミングで実行されたとしても、後続の系列を実行できるような適当な各入力動作の実行タイミングが存在するとき、 w は must トレース可能であるという。一方、 w 中の各出力動作がある適当なタイミングで実行されたとき、後続の系列を実行できる

ような適当な各入力動作の実行タイミングが存在するとき、 w は may トレース可能であるという。これらの概念は、形式的には以下のように定義される。

定義 3.2 シンボリックトレース w に対して、論理式 $TrCondMust(w)$ および $TrCondMay(w)$ を図 2* のように再帰的に定義し、 $TrCondMust(w)$ (あるいは $TrCondMay(w)$) が真であるとき、 w は must トレース可能 (may トレース可能) であるという。

図 2 の $TrCondMust'(w)$ において、もし $a_i \$i$ が出力動作 ($\$i = !$) なら、まず、その出力動作が実行可能となるタイミングが存在することを $\exists t'_i [P_i[t'_i/t_i] \wedge (t \leq t'_i)]$ の部分で要求している。次に $\forall t_i [(P_i \wedge (t \leq t_i)) \Rightarrow TrCondMust'(w, t_i)]$ の部分で、どのような出力タイミング t_i に対しても、後続の系列 w を must トレース可能とすることを要求している。もし、 $a_i \$i$ が入力動作 ($\$i = ?$) なら、後続の系列 w を must トレース可能とするようなタイミング t_i が存在することのみを要求している。

例 3.2 シンボリックトレース $w = (a!, t_a, 1 \leq t_a \leq 5) (b?v_1, t_b, t_b \leq t_a + v_1 \wedge t_b \leq 5)$ に対して、

$$\begin{aligned}
TrCondMust(w) &= \exists v_1 [TrCondMust'(w, 0)] \\
&= \exists v_1 [\exists t'_a [(1 \leq t'_a \leq 5) \wedge (0 \leq t'_a)] \\
&\quad \wedge \forall t_a [(1 \leq t_a \leq 5) \wedge (0 \leq t_a)] \\
&\quad \Rightarrow TrCondMust'((b?v_1, t_b, \\
&\quad \quad t_b \leq t_a + v_1 \wedge t_b \leq 5), t_a)]] \\
&= \exists v_1 [\forall t_a [(1 \leq t_a \leq 5) \\
&\quad \Rightarrow \exists t_b [(t_b \leq t_a + v_1 \wedge t_b \leq 5) \\
&\quad \quad \wedge (t_a \leq t_b) \wedge true]]] \\
&= true
\end{aligned}$$

* 図 2 中の $\Rightarrow$ は含意を表す。すなわち、 $P \Rightarrow Q \stackrel{\text{def}}{=} (\bar{P} \vee Q)$ 。

入力: $w = (a_1 \$1, t_1, P_1) \dots (a_n \$n, t_n, P_n)$

出力: w の must トレース可能性の判定結果と, 可能な場合は各動作の実行可能な時刻範囲

アルゴリズム:

- $TrCondMust_{n+1} := true$
- for $k = n$ downto 1 do
 - if $\$k = ?$ then
 - * 線形不等式の論理積 $P_k(t_1, \dots, t_k) \wedge t_{k-1} \leq t_k \wedge TrCondMust_{k+1}$ を t_k を含む項と含まない項に分けて $Q(t_1, \dots, t_k) \wedge R(t_1, \dots, t_{k-1})$ の形に変形し, $Q(t_1, \dots, t_k) \equiv \bigwedge_{i \in I} \{f_i(t_1, \dots, t_{k-1}) \leq t_k\} \wedge \bigwedge_{j \in J} \{t_k \leq g_j(t_1, \dots, t_{k-1})\}$ なる $\{f_i\}_{i \in I}, \{g_j\}_{j \in J}$ を求める.
 - * $t_k^{inf} := \max\{f_i(t_1, \dots, t_{k-1}) | i \in I\}$
 - * $t_k^{sup} := \min\{g_j(t_1, \dots, t_{k-1}) | j \in J\}$
 - * $TrCondMust_k := [\max\{f_i(t_1, \dots, t_{k-1}) | i \in I\} \leq \min\{g_j(t_1, \dots, t_{k-1}) | j \in J\} \wedge R(t_1, \dots, t_{k-1})]$
 - else
 - * $TrCondMust_{k+1}$ を t_k を含む項と含まない項に分け, $Q(t_1, \dots, t_k) \wedge R(t_1, \dots, t_{k-1})$ の形に変形し, $Q(t_1, \dots, t_k) \equiv \bigwedge_{i \in I} \{f_i(t_1, \dots, t_{k-1}) \leq t_k\} \wedge \bigwedge_{j \in J} \{t_k \leq g_j(t_1, \dots, t_{k-1})\}$ なる $\{f_i\}_{i \in I}, \{g_j\}_{j \in J}$ を求める.
 - * $P_k(t_1, \dots, t_k) \wedge t_{k-1} \leq t_k$ を t_k を含む項と含まない項に分け, $Q'(t_1, \dots, t_k) \wedge R'(t_1, \dots, t_{k-1})$ の形に変形し, $Q'(t_1, \dots, t_k) \equiv \bigwedge_{i \in I'} \{f'_i(t_1, \dots, t_{k-1}) \leq t_k\} \wedge \bigwedge_{j \in J'} \{t_k \leq g'_j(t_1, \dots, t_{k-1})\}$ なる $\{f'_i\}_{i \in I'}, \{g'_j\}_{j \in J'}$ を求める.
 - * $t_k^{inf} := \max\{f'_i(t_1, \dots, t_{k-1}) | i \in I'\}$
 - * $t_k^{sup} := \min\{g'_j(t_1, \dots, t_{k-1}) | j \in J'\}$
 - * $TrCondMust_k := [\max\{f'_i(t_1, \dots, t_{k-1}) | i \in I'\} \leq \min\{g'_j(t_1, \dots, t_{k-1}) | j \in J'\} \wedge \max\{f_i(t_1, \dots, t_{k-1}) | i \in I\} \leq \max\{f'_i(t_1, \dots, t_{k-1}) | i \in I'\} \wedge \min\{g'_j(t_1, \dots, t_{k-1}) | j \in J'\} \leq \min\{g_j(t_1, \dots, t_{k-1}) | j \in J\} \wedge R(t_1, \dots, t_{k-1}) \wedge R'(t_1, \dots, t_{k-1})]$
- $D(v_1, \dots, v_m) := TrCondMust_1$
- $D(v_1, \dots, v_m)$ が充足可能であれば “must-traceable” と判定し, must トレース可能となる入力値に対する条件 $D(v_1, \dots, v_m)$, その際の各動作の実行可能な時刻範囲 $\{(t_k^{inf}, t_k^{sup})\}_{k=1, \dots, n}$ を出力する. 充足不能であれば “not must-traceable” を出力する.

図 3 must トレース可能性判定アルゴリズム

Fig. 3 Algorithm for checking must traceability.

となり, w は must トレース可能である. 一方, シンボリックトレース $w' = (a!, t_a, 1 \leq t_a \leq 6)(b? \mathbf{v}_1, t_b, t_b \leq t_a + \mathbf{v}_1 \wedge t_b \leq 5)$ は $5 < t_a \leq 6$ の時刻 t_a で出力 $a!$ が実行されると入力 $b?$ は実行不可能なため must トレース可能ではないが, $1 \leq t_a \leq 5$ の時刻 t_a で出力 $a!$ が実行されると入力 $b?$ は実行可能であるため, may トレース可能である.

定義 3.3 $TrCondMust(w)$ ($TrCondMay(w)$) の冠頭標準形を

$$\exists \mathbf{v}_1 \dots \exists \mathbf{v}_m$$

$$[Q_1 t_1 \dots Q_n t_n [P(t_1, \dots, t_n, \mathbf{v}_1, \dots, \mathbf{v}_m)]]$$

(ただし, $Q_i \in \{\exists, \forall\}$) とし, $Q_i = \exists$ であるような i の集合を $\{i_1, \dots, i_k\}$ とする. このとき, インターバルの組 $[t_{i_k}^{inf}(t_1, \dots, t_{i_{k-1}}), t_{i_k}^{sup}(t_1, \dots, t_{i_{k-1}})]$ および入力値 $\mathbf{v}_1, \dots, \mathbf{v}_m$ のとりうる値に関する論理式 $D(\mathbf{v}_1, \dots, \mathbf{v}_m)$ が

$$\forall \mathbf{v}_1 \dots \forall \mathbf{v}_m \forall t_1 \dots \forall t_n$$

$$[(D(\mathbf{v}_1, \dots, \mathbf{v}_m) \wedge$$

$$t_{i_1}^{inf} \leq t_{i_1} \leq t_{i_1}^{sup} \wedge \dots \wedge t_{i_k}^{inf} \leq t_{i_k} \leq t_{i_k}^{sup})$$

$$\implies P(t_1, \dots, t_n, \mathbf{v}_1, \dots, \mathbf{v}_m)]$$

を満たすとき, $[t_{i_k}^{inf}(t_1, \dots, t_{i_{k-1}}), t_{i_k}^{sup}(t_1, \dots,$

$t_{i_{k-1}})]$ および $D(\mathbf{v}_1, \dots, \mathbf{v}_m)$ を w の must トレースに関する (may トレースに関する) 解であるという.

3.2 must/may トレース可能性判定アルゴリズム

本節では, 各動作の時間制約が変数の線形不等式の論理積で表されるシンボリックトレースに対し, それらが must/may トレース可能かどうかの判定を行うアルゴリズムを示す. ただし, 簡単のため P における線形不等式 $f(t, x_1, x_2, \dots) < t$ は, 十分小さい正数 ρ を用いて $f(t, x_1, x_2, \dots) + \rho \leq t_n$ のように変換する. これにより, 使用する不等号としては $\leq$ もしくは $\geq$ のみを考える. また, 以降の例においてはこの ρ を省略する.

シンボリックトレース $(a_1 \$1, t_1, P_1) \dots (a_n \$n, t_n, P_n)$ に対する must トレース可能性判定アルゴリズムを図 3 に示す. このアルゴリズムは, 直観的には以下のように説明できる.

最終動作 a_n については後続動作は存在しないので, a_n の実行可能な時刻 t_n は $P_n \wedge (t_{n-1} \leq t_n)$ を満たすような時刻である. 各時間制約 P_k は変数の線形不等式の論理積に限っているため, その時間制約は適当な移項により, 次の 3 つのタイプの等価な線形不等式に置き換えることができる:

(1) $\{f_i(t_1, \dots, t_{n-1}) \leq t_n | i \in I\}$, (2) $\{t_n \leq g_j(t_1, \dots, t_{n-1}) | j \in J\}$, (3) 時間変数 t_n を含まない論理式 $R(t_1, \dots, t_{n-1})$. このため, 時間制約の下限 $t_n^{inf} = \max\{f_i(t_1, \dots, t_{n-1}) | i \in I\}$ と上限 $t_n^{sup} = \min\{g_j(t_1, \dots, t_{n-1}) | j \in J\}$ を式形で求めることができる*. 動作 a_n の実行時刻 t_n が存在するためには, $t_n^{inf} \leq t_n^{sup}$ かつ, $R(t_1, \dots, t_{n-1})$ が成り立たなければならない. すなわち, $TrCondMust_n = [t_n^{inf} \leq t_n^{sup} \wedge R(t_1, \dots, t_{n-1})]$ と表せる. $t_n^{inf} \leq t_n^{sup}$ は $\bigwedge_{i \in I} \bigwedge_{j \in J} [f_i(t_1, \dots, t_{n-1}) \leq g_j(t_1, \dots, t_{n-1})]$ と表せるため, $TrCondMust_n$ には変数 t_n が含まれず, $t_1, \dots, t_{n-1}$ からなる線形不等式の論理積で表現される (動作 a_n が $a_n? \mathbf{v}$ のようにデータ $\mathbf{v}$ を入力する場合, その入力値を表す変数も線形不等式に含まれる可能性がある). $TrCondMust_n$ が真の場合, 動作 a_n の実行可能時刻 t_n は時間範囲 $[t_n^{inf}, t_n^{sup}]$ となる.

次に $\$k = ?$ であるような入力動作 ($k < n$) に関しては, 後続の I/O 動作系列を must トレース可能とするための条件 $TrCondMust_{k+1}$ を考慮して, $P_k \wedge (t_{k-1} \leq t_k) \wedge TrCondMust_{k+1}$ の上界 t_k^{sup} , 下界 t_k^{inf} を式形で計算し, 最終動作 a_n の場合と同様に $TrCondMust_k$ を求める. 入力動作 $a_k?$ の実行可能時刻 t_k は時間範囲 $[t_k^{inf}, t_k^{sup}]$ となる.

また, $\$k = !$ であるような出力動作 ($k' < n$) に関しては次のように $TrCondMust_{k'}$ を求める. まず, 出力動作 $a_{k'}$ が実行可能となるための条件式 $P_{k'} \wedge t_{k'-1} \leq t_{k'}$ を上と同様に計算し, $t_{k'}$ の下界 $t_{k'}^{inf} = l_P(t_1, \dots, t_{k'-1})$, 上界 $t_{k'}^{sup} = u_P(t_1, \dots, t_{k'-1})$, および, $t_{k'}$ に依存しない条件 $R(t_1, \dots, t_{k'-1})$ を構成する. また, 後続動作から繰り越されてきた条件 $TrCondMust_{k'+1}$ についても, 同様に変形を行い, $t_{k'}$ の下界 $l_{TrCondMust}(t_1, \dots, t_{k'-1})$, 上界 $u_{TrCondMust}(t_1, \dots, t_{k'-1})$, および, $t_{k'}$ に依存しない条件 $R'(t_1, \dots, t_{k'-1})$ を構成する. すると, 出力動作 $a_{k'}$ が実行可能で, かつ, $a_{k'}$ の任意の出力タイミング $t_{k'}$ で $a_{k'}$ 以降の I/O 動作列が実行可能であるための条件は, (1) $l_P(t_1, \dots, t_{k'-1}) \leq u_P(t_1, \dots, t_{k'-1})$ であること (出力 $a_{k'}$ が実行可能), かつ, (2) その実行可能時刻の範囲 $[l_P(t_1, \dots, t_{k'-1}), u_P(t_1, \dots, t_{k'-1})]$ が $a_{k'+1}$ 以降の後続動作を must トレース可能とするための $t_{k'}$ の範囲 $[l_{TrCondMust}(t_1, \dots, t_{k'-1}), u_{TrCondMust}(t_1, \dots, t_{k'-1})]$ に含まれていること ($l_{TrCondMust}(t_1, \dots, t_{k'-1}) \leq l_P(t_1,$

$\dots, t_{k'-1}) \wedge u_P(t_1, \dots, t_{k'-1}) \leq u_{TrCondMust}(t_1, \dots, t_{k'-1})$), および, (3) 論理式 $R(t_1, \dots, t_{k'-1})$, $R'(t_1, \dots, t_{k'-1})$ を満たすこと, と同値となる. この条件を $TrCondMust_{k'}$ として先行動作の条件へ繰り越す.

このように再帰的に計算を行い $TrCondMust_1$ を求める. $TrCondMust_1$ は $a_1, \dots, a_n$ で入力される値を表す変数 $\mathbf{v}_1, \dots, \mathbf{v}_m$ と V に含まれる変数の初期値からなる不等式の論理結合で表される. これらの式に変数の初期値を代入すると, $\mathbf{v}_1, \dots, \mathbf{v}_m$ のみからなる不等式の論理結合 $D(\mathbf{v}_1, \dots, \mathbf{v}_m)$ が得られる. もしこの論理式が充足可能であれば, 与えられたシンボリックトレースは must トレース可能であると結論できる. その充足可能性は線形計画法を用いて判定できる. もし充足可能であれば, 充足可能となるための具体的な $\mathbf{v}_1, \dots, \mathbf{v}_m$ の値も求めることができる.

なお, 一般に上述の $l_{TrCondMust}(t_1, \dots, t_{k'-1}) \leq l_P(t_1, \dots, t_{k'-1})$ は $\max\{f_i(t_1, \dots, t_{k'-1}) | i \in I\} \leq \max\{f'_i(t_1, \dots, t_{k'-1}) | i \in I'\}$ の形で表されるが, この式は $\max\{f_1, \dots, f_p\} \leq f'_1 \vee \max\{f_1, \dots, f_p\} \leq f'_2 \vee \dots \vee \max\{f_1, \dots, f_p\} \leq f'_q$ のように q 通りに場合分けして考える必要がある. q 通りの各場合については, $\max\{f_1, \dots, f_p\} \leq f'_j$ を $\bigwedge_{i=1, \dots, p} (f_i \leq f'_j)$ のように不等式の論理積で表すことができる.

一般には, 上述のように出力動作の部分で場合分けが生じるため, 求めたいテスト系列に含まれる出力動作の数の指数オーダーに比例する数の場合分けが必要となる. 求めるテスト系列の長さがそれほど長くない場合は, この場合分けの数はあまり問題にならないが, 場合分けが多くなり, 充足可能性の判定を高速に行いたい場合, たとえば, 次のような十分条件で判定する方法が考えられる. その十分条件では, $\max\{f_i(t_1, \dots, t_{k'-1}) | i \in I\} \leq \max\{f'_i(t_1, \dots, t_{k'-1}) | i \in I'\}$ の部分を $\max\{f_i(t_1, \dots, t_{k'-1}) | i \in I\} \leq \min\{f'_i(t_1, \dots, t_{k'-1}) | i \in I'\}$ に変更した式を用いて $TrCondMust$ を求める. もし, その式が充足可能であれば, もとの $TrCondMust$ も充足可能となり, must トレース可能となる. この場合, 変更した式は $\bigwedge_{i=1, \dots, p} \bigwedge_{j=1, \dots, q} (f_i(t_1, \dots, t_{n-1}) \leq f'_j(t_1, \dots, t_{n-1}))$ のように論理和を含まない1つの論理積の形で表せるため, 場合分けの必要がなくなり, $TrCondMust$ の充足可能性の判定を高速に行える. ただし, 十分条件を用いているため, 試験の可能性は狭くなる.

例として図4の系列に対し, must トレース可能性判定のアルゴリズムを適用すると, 図5のようにな

* ここで, I, J はそれぞれ (1), (2) のタイプの線形不等式中の f_i, g_j の添字 i, j の集合を表す.

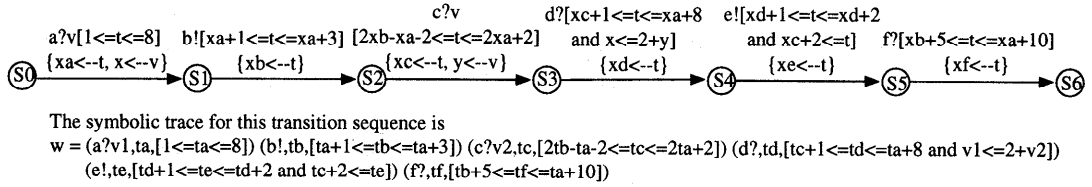


図 4 系列の例

Fig. 4 A transition sequence.

$$\begin{aligned}
 (t_f^{inf}, t_f^{sup}) &= (\max\{t_e, t_b + 5\}, t_a + 10) \\
 TrCondMust_f &= [\max\{t_e, t_b + 5\} \leq t_a + 10] \\
 (t_e^{inf}, t_e^{sup}) &= (\max\{t_d, t_d + 1, t_c + 2\}, t_d + 2) \\
 TrCondMust_e &= [\max\{t_d, t_d + 1, t_c + 2\} \leq t_d + 2 \wedge t_d + 2 \leq t_a + 10 \wedge t_b + 5 \leq t_a + 10] \\
 (t_d^{inf}, t_d^{sup}) &= (\max\{t_c, t_c + 1\}, \min\{t_a + 8, t_a + 10\}) \\
 TrCondMust_d &= [\max\{t_c, t_c + 1\} \leq \min\{t_a + 8, t_a + 10\} \wedge t_b + 5 \leq t_a + 10 \wedge v_1 \leq 2 + v_2] \\
 (t_c^{inf}, t_c^{sup}) &= (\max\{t_b, 2t_b - t_a - 2\}, \min\{2t_a + 2, t_a + 7\}) \\
 TrCondMust_c &= [\max\{t_b, 2t_b - t_a - 2\} \leq \min\{2t_a + 2, t_a + 7\} \wedge t_b + 5 \leq t_a + 10 \wedge v_1 \leq 2 + v_2] \\
 (t_b^{inf}, t_b^{sup}) &= (\max\{t_a, t_a + 1\}, t_a + 3) \\
 TrCondMust_b &= [\max\{t_a, t_a + 1\} \leq t_a + 3 \wedge t_a + 3 \leq \min\{t_a + 5, 2t_a + 2, t_a + 7, 1.5t_a + 2, t_a + 4.5\} \\
 &\quad \wedge v_1 \leq 2 + v_2] \\
 (t_a^{inf}, t_a^{sup}) &= (\max\{1, 2\}, 8) \\
 TrCondMust_a &= [\max\{1, 2\} \leq 8 \wedge v_1 \leq 2 + v_2]
 \end{aligned}$$

図 5 must トレース可能性の判定

Fig. 5 Checking must traceability.

る。よって、入力動作 $a?v_1$, $c?v_2$ で $v_1 \leq 2 + v_2$ を満たすような v_1 , v_2 を入力値として与えるとき、この系列は must トレース可能である。また、この結果から分かるように、各入出力動作の実行可能な時間範囲は、それに先行する入出力動作の実行時刻を表す変数の関数表現として求めることができる。さらに、実際に系列を実行する際の入力動作 $a_i?$ については、時間幅 $[t_i^{inf}, t_i^{sup}]$ の中から任意の実行タイミングを選ぶことができる（出力の場合、その時間幅で出力されることを確認する）。その実行タイミング t_i' が決まると、以降の動作の実行可能な時間範囲を表す関数に含まれる $a_i?$ の実行時刻を表す変数 t_i に具体値 t_i' を与えて部分計算することにより、後続の入出力動作の実行可能な時間範囲が順次確定していく。

なお、may トレース可能性判定アルゴリズムは図 3 中の各 $TrCondMust$ を $TrCondMay$ に置き換え、各出力動作を入力動作として取り扱う（「if $\$k = ?$ then ...」文を then 文の中身だけに置き換える）ことにより求めることができるので、ここでは詳細は省略する。may トレース可能性の利用法であるが、たとえば、先行する出力動作が指定された時間の後半に実行されたときのみ後続の動作が実行可能な場合（たとえばタイムアウトの特別な場合など）、must トレース可能性は成立しない。しかし、そのような場合でも、先行する出力動作がどの時間範囲に実行されたら後続の動作が

実行可能となるのかなどが分かれば試験が容易になる。may トレース可能性はそのような場合に利用することができる。

4. I/O 時間オートマトンに対する適合性試験

ここでは UIOv-法をもとに 2 章で提案した I/O 時間オートマトンに対する適合性試験法を提案する。

4.1 仕様と IUT に対する仮定

一般に試験による時間オートマトンの等価性の判定は決定不能であることが知られている³⁾。このため、IUT にいくつかの仮定を与えない限りその正しさを論理的に保証することができない。このため、本論文では仕様と IUT に対して次のような仮定をおき、そのもとで実装の正しさを議論する。

- 仕様は、時間制約を無視したオートマトンと見なした場合に、完全・最小の決定性 FSM である。
- 十分大きな時間間隔 L に対して、仕様の各状態からの出力遷移はその状態に遷移してから L 単位時間以内に実行されるように記述されている。
- 仕様と IUT の入出力動作集合は同じである。
- 仕様、IUT とともに誤りのないリセット遷移がある。
- IUT は仕様の状態数を超えない I/O 時間オートマトンとして実装されている。
- IUT から同一の入出力動作系列が観測される場合、その実行時刻にかかわらず到達する状態はただ 1

つに定まる*。

- 試験者が与えた入力 IUT に受け付けられたか (入力動作が実行されたか) どうかを, 何らかの方法で試験者が確認できる。

- 各シンボリックトレース w に対して, w の入力動作系列を異なる実行タイミングで十分大きな整数 F 回以上 IUT に与えたあと, 入力動作 $a?$ が実行できないことを確認した (あるいは出力動作 $b!$ が観測されない) 場合, その w の実行により遷移する IUT の状態からは入力動作 $a?$ (あるいは出力動作 $b!$) は実行できないものとする**。

4.2 提案する適合性試験法

以下では, 上述のような仮定のもとで, (1) 与えられた仕様の各状態と各遷移が正しく実装されているかどうかを試験した後, (2) 各遷移条件の述語が正しく実装されているかどうかを試験する。

4.2.1 状態と遷移の実装の正しさ

提案する手法では, 次のような順に状態確認を行う。

- (1) 仕様 M から時間制約を取り除いた FSM を構成し, その FSM の各状態 s_i に対する UIO 系列 u_i ($i = 0, \dots, n$) の集合 $U = \{u_1, \dots, u_n\}$ を求める。また, 初期状態から各状態 s_i へ遷移させるような先行系列 v_i の集合 $V = \{v_1, \dots, v_n\}$ を求める。
- (2) 先行系列の集合 V と UIO 系列の集合 U から各状態への先行系列とその状態の UIO 系列の連結集合 $V.U_0 = \bigcup_i v_i.u_i$ を構成する。また, 各状態 s_i の先行系列とその状態の UIO 系列ではない UIO 系列 u_j ($i \neq j$) の連結 $v_i.u_j$ (仕様上でその I/O 動作系列すべては実行できない) について, 実際にその系列が仕様上で実行できるまでの部分動作系列 $v_i.u'_j$ と, それに引き続く仕様では存在しない I/O 動作 a_{ij} のみを取り出し, $V.U_x = \bigcup_{i \neq j} v_i.u'_j.a_{ij}$ を構成する。
- (3) 系列集合 $V.U_0$ 中のすべての系列に対して must トレース可能性を判定する。すべてが must トレース可能であれば, これらの系列を試験系列と考える。もし, すべての系列が must トレース可能にならなければ, (1) に戻って新たに異なる U, V を見つける。
- (4) 生成した試験系列 $V.U_0, V.U_x$ を IUT に与える。このとき $V.U_0$ 中の試験系列の各入力は, 3 章で

求めた must トレースの解である関数表現を用いて実際の出力動作の実行時刻から計算したタイミングで実行し, 出力に関しては仕様で記述された時間範囲で実行されたかどうかを確認しながら試験する (前節の仮定より, たかだか L 単位時間以内に出力が観測される)。 $V.U_x$ 中の各試験系列 $v_i.u'_j.a_{ij}$ については, 十分大きな整数 F 回以上異なる実行タイミングで $v_i.u'_j$ を実行したあと, a_{ij} が実行できないことを確認する (u'_j の途中で実行できないことが確認されてもよい)。

(5) 仕様どおりの反応 ($V.U_0$ に対しては実行でき, $V.U_x$ に対しては a_{ij} が実行できないこと) を IUT から観測できれば, 状態確認を行えたと判断する。

また, 同様にして遷移確認も行う。その際は, 状態確認の際に用いたものと同じ先行系列の集合 V , 入出力記号 A , UIO 系列の集合 U を連結させた系列集合 $V.A.U_0 = \{v_i.a_{ij}.u_j | a_{ij} \text{ は状態 } s_i \text{ からの遷移}\}$, $V.A_x = \{v_i.a_{ij} | a_{ij} \text{ は状態 } s_i \text{ からの遷移として存在しない遷移}\}$ を構成し, $V.A.U_0$ については仕様どおりの反応が得られること, $V.A_x$ については十分大きな整数 F 回以上異なる実行タイミングで先行系列 v_i を実行したあとつねに a_{ij} が実行できないことを確認する。仕様どおりの反応が IUT から観測できれば, 遷移確認を行えたと判断する。

試験系列 $V.U_0, V.U_x$ を IUT に与え, 仕様どおりの反応を確認することにより, 仕様から得られた UIO 系列の集合 U が IUT の n 個の状態を区別するための UIO 系列の集合にもなっていることが保証できる。遷移確認の際に状態確認の際に用いたものと同じ先行系列の集合 V を用いることで, 各遷移の開始状態が仕様上でその遷移の開始状態に対応していることが保証できる。また, 各遷移を実行したあとの状態に UIO 系列を与えることにより, 遷移の実行後の状態が仕様と同じであることを確認できる。

4.2.2 遷移条件の実装の正しさ

以下では, 遷移条件の実装の誤りとして予想できる誤りをいくつか特定し, 特定した誤りに対して, その誤りを検出するための一手法を提案する。

遷移条件の実装の正しさの試験は初期状態に近い遷移から順に行う。初期状態から $k-1$ 個の遷移を実行した後実行される k 番目の遷移の遷移条件の実装の正しさを試験する際は, 先行する $k-1$ 個の遷移の遷移条件の実装の正しさは試験済みであるとして議論を進める。簡単のため, 以下では k 番目の遷移の遷移条件が $(A_1 \leq t_k) \wedge (A_2 \leq t_k) \wedge \dots \wedge (A_n \leq t_k) \wedge (t_k \leq B_1) \wedge (t_k \leq B_2) \wedge \dots \wedge (t_k \leq B_m)$ の形の論理積で表され, そのうちの 1 つ, たとえば, $(A_n \leq t_k)$ に

* この条件は IUT が時間決定的に動作することを要求している。
 IUT が非決定的に動作する場合, 文献 8) のような非決定性のオートマトンに基づく試験法を用いる必要がある。

** 本来は, 可能なすべての実行タイミングで w を実行したあと動作 a を実行できないことを示す必要があるが, すべての可能な実行タイミングを求めることも試験することも不可能であるので, ここでは, F 回以上 w を IUT に与えたあと入力動作 $a?$ が実行できないとき, $a?$ は実行不可能であると判断している。

(1) $(A_h < t_k)$, (2) $(A_h \geq t_k)$, (3) $(A_h > t_k)$, (4) $(A_h = t_k)$, (5) $(A_h + C \leq t_k)$, (6) $(A_h - C \leq t_k)$ の誤りがただか 1 つある場合を想定する (C は正定数)。

3章で述べたように、上述の条件を満足する試験系列は $\max(A_1, \dots, A_n) \leq t_k \leq \min(B_1, \dots, B_m)$ を満たす時間に k 番目の遷移を実行できる。このとき、もし A_h が $\max(A_1, \dots, A_n)$ より小さければ $(A_h \leq t_k)$ が $(A_h < t_k)$ のように誤って実装されていても何の影響もない。たとえば、 $(t_{k-1} + 1 \leq t_k) \wedge (t_{k-1} < t_k)$ なる仕様上の遷移条件が、 $(t_{k-1} + 1 \leq t_k) \wedge (t_{k-1} < t_k)$ のように誤って実装されていたとしても、この述語は $(t_{k-1} + 1 \leq t_k)$ と等価なため、誤った実装 $(t_{k-1} < t_k)$ の部分が全体の遷移条件の真偽に影響しない。このような遷移条件の実装の正しさは試験できないので、以下では任意の A_p に対して $A_p + \epsilon < A_h$ (ϵ は十分小さい正定数, $\epsilon \ll C$) なる条件を加えても実行可能であるような遷移条件に対してのみその実装の正しさの試験を行う。

k 番目の遷移の実行時刻として、(a) $\max(A_1, \dots, A_n) - \epsilon$, (b) $\max(A_1, \dots, A_n)$, (c) $\max(A_1, \dots, A_n) + \epsilon$ のようなものを選ぶことにする。正しく実装されている場合と上記の (1)~(6) の誤りがある場合についての反応を比較すると次のようになる (○印は実行可能, ×印は実行不可能)。

反応	(a)	(b)	(c)
仕様	×	○	○
(1)	×	×	○
(2)	○	○	×
(3)	○	×	×
(4)	×	○	×
(5)	×	×	×
(6)	○	○	○

上の表から明かなように、(1)~(6) の誤りがある場合は仕様の反応と異なる反応を返すため、その誤りを検出できる。上の例では、 k 番目の遷移までのシンボリックトレース w を作成し、 k 番目の遷移が入力動作なら、上記の時刻 (b), (c) で k 番目の入力を実行できることを確認するとともに、異なる実行タイミングで F 回以上 w を実行し、そのいずれの場合にも k 番目の入力動作が時刻 (a) で実行できないことを確認できれば、 k 番目の遷移の遷移条件の実装が正しいことが分かる。 k 番目の遷移が出力動作なら、十分な回数 w を実行し、時刻 (b), (c) で k 番目の出力が実行されることを確認するとともに、時刻 (a) で k 番目

の出力が実行されないことを確認することにより、遷移条件の実装が正しさが保証される。

上述 (1)~(6) 以外の誤りがある場合や上記の実装の誤りが 2 つ以上ある場合などを考慮すると、上述のように単純にはその誤りを検出できない。このように、一般には (a)~(c) の試験のみでは十分とはいえないが、(1)~(6) のような誤りは典型的誤りとも考えられるので、このような試験を行うことにより、かなりの遷移条件の誤りが検出できると思われる。

5. 本試験系列生成法の適用例

図 1 のモデルに対して、提案する手法を適用してみよう。この例に対する UIO 系列の集合 U は、たとえば、

$$\begin{aligned}
 U = \{ & s_0 : \text{data_start? data_end? first_display_start!}, \\
 & s_1 : \text{data_end? first_display_start!}, \\
 & s_2 : \text{first_display_start!}, \\
 & s_3 : \text{first_display_end!}, \\
 & s_4 : \text{data_start? data_end? display_start!}, \\
 & s_5 : \text{data_end? display_start!}, \\
 & s_6 : \text{display_start_intime!}, \\
 & s_7 : \text{display_end! not_mdf!}, \\
 & s_8 : \text{display_end! data_start?}, \\
 & s_9 : \text{not_mdf!} \}
 \end{aligned}$$

また、先行系列集合 V は初期状態から各状態への最短なパスを選ぶ。この V, U に対して must トレース可能な試験系列集合 $V.U_\circ, V.U_\times$ (状態確認) と $V.A.U_\circ, V.A_\times$ (遷移確認) を生成できることを確認した。たとえば、状態 s_9 の確認のための試験系列 $w = \text{data_start? } v_1 \text{ data_end? first_display_start! first_display_end! data_start? } v_2 \text{ data_end? display_start! display_end! not_mdf!}$

が must トレース可能であるための条件は、 $D(v_1, v_2) = [b \leq T_{h2} + v_2 - v_1 - x]$ となる。たとえば、定数 b, T_{h2}, x の値がそれぞれ 10, 2, 10 であったとすると、 $10 \leq 2 + v_2 - v_1 - 10$, すなわち、 $18 \leq v_2 - v_1$ を満たす入力 v_1, v_2 を与えると、 w は must トレース可能になる。また、 w 中の各入力動作の実行可能な時間幅もその入力動作に先行する動作の実行時刻を表す変数と v_1, v_2 を用いた関数で表現できる。

図 1 以外の例として、イーサネットのタイムアウトメカニズムや、映像、音声、文字データの 3 つのメディアデータを可変長パケットに多重化して送信するパケット多重化方式 H.223⁶⁾ などいくつかの実時間プロトコルの仕様が提案する I/O 時間オートマトンでモデル化でき、試験系列も生成できる。

6. あとがき

本論文ではあるクラスの I/O 時間オートマトンを提案し、そのモデルに対して実行タイミングを含む適合性試験系列を自動生成する 1 つの方法を提案した。

仕様や IUT に対する仮定をゆるめることや、遷移条件の実装の正しさを保証することによりより強力な試験法を考案することなどが今後の課題である。

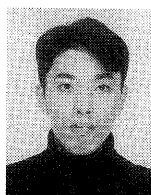
参 考 文 献

- 1) Alur, R. and Dill, D.L.: A theory of timed automata, *Theoretical Computer Science*, Vol.126, pp.183-235 (1994).
- 2) Chanson, S. and Zhu, J.: Automatic Protocol Test Suite Derivation, *Proc. INFOCOM'94*, pp.792-799 (1994).
- 3) Clarke, D. and Lee, I.: Automatic Generation of Tests for Timing Constraints from Requirements, *Proc. 3rd Int. Workshop on Object-Oriented Real-Time Dependable Systems* (1997).
- 4) Higashino, T. and Bochmann, G.v.: Automatic Analysis and Test Case Derivation for a Restricted Class of LOTOS Expressions with Data Parameters, *IEEE Trans. Soft. Eng.*, Vol.20, No.1, pp.29-42 (1994).
- 5) Ishibashi, Y., Tasaka, S. and Minami, E.: Performance measurement of a stored media synchronization mechanism: Quick recovery scheme, *Proc. GLOBECOM'95*, pp.811-817 (1995).
- 6) ITU-T: Multiplexing protocol for low bit rate multimedia communication, Recommendation H.223 (1996).
- 7) Lee, D. and Yannakakis, M.: Principles and Methods of Testing Finite State Machines - A survey, *Proc. IEEE*, Vol.84, No.8, pp.1090-1123 (1996).
- 8) Luo, G., Bochmann, G.v. and Petrenko, A.: Test Selection Based on Communicating Non-deterministic Finite State Machines using a Generalized Wp-method, *IEEE Trans. Soft. Eng.*, Vol.20, No.2, pp.149-162 (1994).
- 9) Mandrioli, D., Morasca, S. and Morzenti, A.: Generating Test Cases for Real-Time Systems from Logic Specifications, *ACM Trans. Computer Systems*, Vol.13, No.4, pp.365-398 (1995).
- 10) Springintveld, J., Vaandrager, F. and D'Argenio, P.R.: Testing Timed Automata, Technical Report, CTIT 97-17, University of Twente (1997).

- 11) Vuong, S.T., Chan, W.L. and Ito, M.R.: The UIOv-Method for Protocol Test Sequence Generation, *Proc. 2nd IFIP Workshop on Protocol Test Systems (IWPTS'89)*, pp.161-175 (1989).
- 12) Wang, C.-J. and Liu, M.T.: Axiomatic Test Sequence Generation for Extended Finite State Machines, *Proc. 12th Int. Conf. on Distributed Computing Systems (ICDCS-12)*, pp.252-259 (1992).

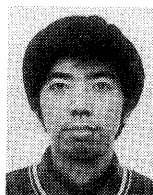
(平成 10 年 4 月 6 日受付)

(平成 10 年 9 月 7 日採録)



深田 敦史

平成 10 年大阪大学大学院基礎工学研究科物理系情報工学分野博士前期課程修了。工学修士。同年ヤマハ(株)勤務。在学中、通信プロトコルの適合性試験に関する研究に従事。



中田 明夫 (正会員)

平成 4 年大阪大学基礎工学部情報工学科卒業。平成 9 年同大学院基礎工学研究科博士後期課程修了。工学博士。現在、広島市立大学情報科学部情報数理学科助手。分散システム、プロセス代数、時相論理等に関する研究に従事。



東野 輝夫 (正会員)

昭和 54 年大阪大学基礎工学部情報工学科卒業。昭和 59 年同大学院基礎工学研究科博士後期課程修了。工学博士。現在、同大基礎工学部情報科学科助教授。平成 2 年、6 年モントリオール大学客員研究員。分散システム、通信プロトコル等に関する研究に従事。IEEE Senior Member。



谷口 健一 (正会員)

昭和 40 年大阪大学工学部電子工学科卒業。昭和 45 年同大学院基礎工学研究科博士課程修了。工学博士。現在、同大学基礎工学部情報科学科教授。この間、計算理論、ソフトウェアやハードウェアの仕様記述・実現・検証の代数的手法および支援システム、関数型言語の処理系、分散システムや通信プロトコルの設計・検証法等に関する研究に従事。