Vol. 35 No. 12 情報処理学会論文誌 Dec. 1994

時間ステートチャートに基づくリアルタイムシステム検証方式

山根 智†

リアルタイムソフトウェアは多くのプロセスが並行動作してタイミング制約が厳しい．つまり，プロセスの論理動作の正当性だけでなく，動作が正しいタイミングで実行されるかどうかが非常に重要である．従来より，構造化分析やペトリネット，オートマトン，プロセス代数，時相論理，モードチャートなどが研究されている．これらの手法は，形式性の欠如や状態と動作の統一的記述の困難，直感的な理解性の欠如，プロセス合成の困難，検証アルゴリズムの複雑さ，などの問題点がある．本稿では，状態と動作の統一的記述や検証アルゴリズムの容易性，形式的なドキュメント性，タイミング検証などを重視して，時間ステートチャートに基づく仕様記述と検証の手法を提案する．本手法の主要な特徴は次のとおりである：(1)時間ステートチャートは時間 Buchi オートマタの階層化 (OR-分割) と並行化 (AND-分割) から構成する．(2)検証性質記述は決定性時間 Muller オートマトンから構成する．(3)検証アルゴリズムは時間 Buchi オートマタと決定性時間 Muller オートマトンの (形式言語理論における) 言語包合問題に帰着する．最後に，事例を通して本手法の有効性を示す．

Method of Verification of Real-Time System Based on Timed Statechart

SATOSHI YAMANE†

Many processes concurrently behave and timing constraint is strict in real-time software. That is to say, it is very important for processes to behave on valid timing condition as well as compute valid values. Many research have been done in structured analysis and Petri-Net, automaton, process algebra, temporal logic, modechart. There are many problems, for example, lack of formalizm, difficulties in uniformly specifying behavior and states, lack of intuitonic understandability, difficulties in composition of processes, complexity of verification algorithm. In this paper, we propose the method of specification and verification based on timed statechart. We aim at uniformly specifying behavior and states, simplification of verification algorithm, formal documentability, timing verification. The main feature is following: (1) Timed statechart consists of hierarchy (OR-decomposition) and concurrency (AND-decomposition) of timed Buchi automata. (2) Verification property specification consists of deterministic timed Muller automaton. (3) Verification algorithm reduces inclusion problem between timed Buchi automata and deterministic timed Muller automaton in formal language theory. Finally, we show this method effective using example.

1. はじめに

航空機や家電製品などのリアルタイムソフトウェアは，多くのプロセスが並行動作してタイミング制約が厳しい．つまり，プロセスの論理動作の正当性だけでなく，動作が正しいタイミングで行われるかどうかが非常に重要である．このために，タイミング制約記述を含む適切な仕様記述と検証の手法が必要である[1]．

従来より，リアルタイムソフトウェアの仕様記述手法としては，構造化分析やペトリネット，オートマトン，プロセス代数，時相論理，モードチャートなどが研究されている．

①構造化分析[2]は，ドキュメント性が高く，米国国防省を中心に実用化されているが，形式性が弱く，検証能力に問題がある．

②モードチャート[3]も形式性が弱い点では同様であり何らかの形式的モデルの導入が必要であり，それ自体独立で使用できるものではない．

③ペトリネット[4]は並行プロセスの記述や解析の能力は高いが，トランジションやプレースの中のトークンの動作としてモデル化しなければならず，抽象度

† 島根大学理学部情報科学科
Department of Computer Science, Faculty of Science, Shimane University

や直観性が低く，決して使いやすい仕様記述手法ではない．

④プロセス代数[5),6)]は，観測等価性による仕様と実装の検証がポイントであり，検証を考慮した仕様記述手法としては望ましい．しかし，動作の順序関係に着目したモデル化であるため，状態の概念が扱いにくく，並行プロセスの合成作業も容易でない．

⑤時相論理[7)]は，健全で完全な公理系のもとで，仕様を時相論理式で記述して，定理証明アルゴリズムで検証するものがある．また，システム仕様が時相論理式を充足するかどうかを判定するモデルチェッキングによる検証もある．しかし，動作の結果生じた状態にあるとき真となるような命題変数を考え，命題変数を時相論理式として扱うために，動作が理解しにくい．

⑥オートマトン[8)]は，状態と動作を統一的に扱いながら形式言語理論の成果のもとで，言語の包合問題として検証が実現できる．しかし，並行プロセスが記述できないことや複雑なプロセスの状態遷移図が記述しにくいといった問題がある．

のように，従来の仕様記述手法では，形式性の欠如や状態と動作の統一的記述の困難さ，プロセス合成の困難さ，形式的なドキュメント性の欠如，検証アルゴリズムの複雑さなどの問題点がある．

一方，上記問題点を解決すべくオブジェクトチャート[9)]が提案されている．本手法は，オブジェクト指向分析の中の動作モデルをステートチャート[10)]で拡張したものであり，次の特徴を有する：

①VDM (Vienna Development Method) の論理体系である LPF (Logic of Partial Functions)[11)]により状態遷移仕様を形式化して検証可能としたこと

②3項組（送信オブジェクト，サービス要求，受信オブジェクト）により，オブジェクト動作を定義したこと

しかし，リアルタイムソフトウェアの仕様記述と検証に対しては，①と②の各々には次の問題点がある：

①VDM の仕様記述と検証はプログラムの停止性を前提としている[11)]．しかし，リアルタイムソフトウェアは停止しないプログラムと見なされており[12)] VDM はほとんど役に立たない

②3項組の動作列によりオブジェクトチャートの動作的意味を定義している．また，サービス要求の項にタイマ記述を許してタイミング制約記述を可能としている．しかし，仕様（オブジェクトチャート）が検証性質（動作列）を満たすかどうかの検証手法は提案されていない

そこで，本稿では，上記問題点を解決すべく，時間ステートチャートに基づくリアルタイムソフトウェアの仕様記述と検証の手法を提案する．その主要な特徴は次のとおりである：

①時間ステートチャート（システム仕様）は，時間 Buchi オートマタの階層化（AND-分割）と並行化（OR-分割）から構成する

②検証性質記述は決定性時間 Muller オートマトンから構成する

③①と②よりシステム仕様と検証仕様の統一的な記述を可能とする

④検証アルゴリズムは，時間 Buchi オートマタと決定性時間 Muller オートマトンの言語包合問題に帰着して，タイミング検証を可能とする

以下，2章ではステートチャートや時間オートマトンの諸定義を行い，3章では時間ステートチャートを提案して構文や意味，検証手法を定義する．また，4章では例題により時間ステートチャートの有効性を示して，5章ではまとめを述べる．

2. 諸 定 義

2.1 ステートチャート

従来より，リアルタイムシステムを記述するために有限オートマトンが広く使用されている．しかし，状態数が非常に多いときは，有限オートマトンをダイヤグラム的に記述することが困難である．そこで，ステートチャートは有限オートマトンを拡張して，状態を繰り返し分解しながら AND/OR 形式のサブ状態に展開できるようにした．これにより，仕様記述のモジュール化や階層化，並行化を実現した[10)]．

また，ステートチャートはタイミング記述などが拡張されている．しかし，ステートチャートの検証方法はシミュレーション方式であり，入力されたテストパターンに対する動作以外については正当性が示されないという問題点がある．

以下，階層化や並行化，ブロードキャスト通信などのステートチャートの主要な特徴を説明する．

（1） 階層化 (OR 分割)

階層化は状態をクラスタ化して，状態と状態集合を閉じた等高線により表現する．状態の等高線の境界で作られる遷移はすべての状態集合に適用される．**図1**（a）で示すように，イベントbは状態Aまたは状態C

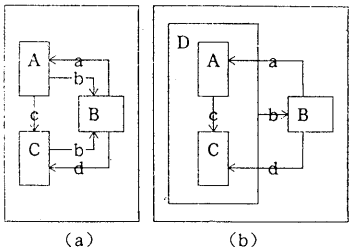


図 1 ステートチャート（階層化）
Fig. 1 Statechart (hierarchy).

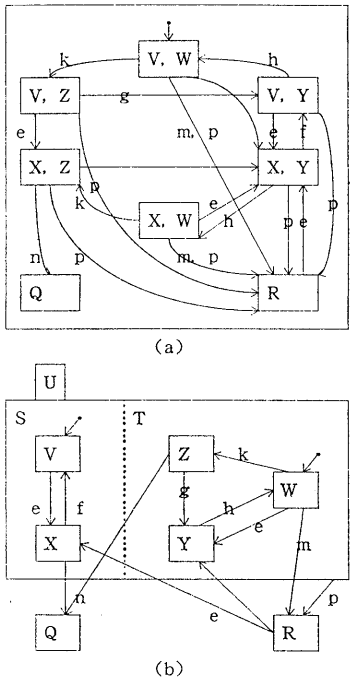


図 2 ステートチャート（並行化）
Fig. 2 Statechart (concurrency).

から状態Bに状態遷移させる．階層化は(b)のようにAとCをDに抽象化して，Dの境界から2つのイベントbを1つのイベントbで置換する．AとCの結合Dは exclusive-OR であり，Dに入ると，AまたはCに遷移する．

（2） 並行化（AND 分割）

AND 分割される状態は AND 構成要素のサブ状態のすべてからなり，AND 分割はサブ状態のカルテジアン積として表現される．**図2**(a)の状態図は(b)と等価であり，(b)のSとTの両方が並行動作する．また，(a)と(b)の相互生成アルゴリズムがある．状態集合を {S, T} で表現すると初期状態は {V, W} である．この状態からイベントeが生起すると {X, Y} に遷移して，イベントkが生起すると {V, Z} に遷移する．このような AND 形式の分解により直交状態ができて，状態の膨張現象が解決でき，独立した並行な状態が記述できる．

（3） ブロードキャスト通信

上記(2)のイベントによる直交状態のすべての遷移は，ブロードキャスト通信により実現される．これは，外部のイベントの生起はすべての直交状態の遷移を引き起こすということである．

2.2 時間オートマトン

時間オートマトン[13),14)]は，リアルタイムシステムや論理回路の仕様記述と検証のために，ω-オートマトンを拡張したものである．これらは，停止性が仮定できない非同期システムである．

（1） ω-オートマトン

ω-オートマトンは，無限語 ω-正規言語を受理する[15),16)]．代表的な ω-オートマトンは，Buchi オートマトンや Muller オートマトンなどであり，ω-正規言語のクラスはブール演算のもとで閉じている．

①Buchi オートマトン

Buchi オートマトン[14)]は，入力集合が無限であり受理状態集合が走査の中で無限に繰り返されるとき受理される．ここで，走査とは入力集合による無限な状態遷移系列をいう．

②Muller オートマトン

Muller オートマトン[14)]は，入力集合が無限であり受理状態集合が状態のベキ集合であり，走査の中で無限に繰り返されるとき受理される．

（2） 時間オートマトン

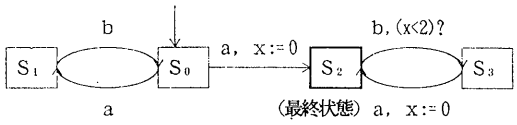
①時間オートマトンは，語とその生起時間の順序対である時間付き言語を受理する．また，グローバル時間により刻まれるクロックを付加した状態遷移表に従って動作する．クロックの表現形式は制約式とリセット式である．なお，制約式 δ は，

$$\delta := x \leqq c \mid c \leqq x \mid \neg\delta \mid \delta_1 \wedge \delta_2$$

である．

②時間 Buchi オートマトンは，和演算と積演算で閉じており，補集合演算では閉じていない．また，言語の空問題は PSPACE である．しかし，言語の包合問題は決定不能である．**図3**に事例を示す．

③決定時間 Muller オートマトンは，ブール演算のもとで閉じている．このために，時間オートマトンの受理言語が決定性時間 Muller オートマトンの受理

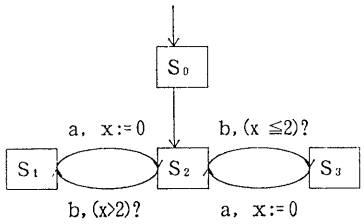


ただし，S_2 を無限に繰り返す走査列が受理される
（a） 時間 Buchi オートマトン
（a） Timed Buchi automaton

$$L=\{((ab)^{\omega},\tau) \mid \exists i.\forall j\geq i.(\tau_{2j}<\tau_{2j-1}+2)\}$$

（b） （a）の受理言語
（b） Acceptance language of (a)

図 3 時間 Buchi オートマトンの例
Fig. 3 Example of timed Buchi automaton.



ただし，受理集合 $\{S_2, S_3\}$ を無限に繰り返す走査列が受理される
（a） 決定性時間 Muller オートマトン
（a） Deterministic timed Muller automaton

$$L=\{((ab)^{\omega},\tau) \mid \exists i.\forall j\geq i.(\tau_{2j+2}\leq\tau_{2j+1}+2)\}$$

（b） （a）の受理言語
（b） Acceptance language of (a)

図 4 決定性時間 Muller オートマトンの例
Fig. 4 Example of deterministic timed Muller automaton.

言語に含まれるかという含合問題は決定可能であり，PSPACE 完全である．**図 4** に事例を示す．

（3） 検 証 方 式

上記時間オートマトンの理論のもとに検証方式が提案されている[14]．

①システム記述

リアルタイムシステムは複数のプロセスの相互作用として表現できるので，積の演算で閉じている必要がある．また，各々のプロセスは時間付き遷移を有する．そこで，リアルタイムシステムの各々のプロセスは時間 Buchi オートマトンでモデル化できて，システムはその積で表現できる．

②検証性質記述

検証性質として，活性や安全性が考えられ，状態のベキ集合の繰り返しや時間付き遷移の表現が必要である．また，検証のためには補集合演算で閉じている必要もある．そこで，決定性時間 Muller オートマトンにより検証性質を記述する．

③検証アルゴリズム

検証問題は，時間 Buchi オートマトンと決定性時間 Muller オートマトンとの言語含合問題に帰着する．時間 Buchi オートマトンと決定性時間 Muller オートマトンの補集合との積が空であれば，対象とする性質を満たすとする．

3. 時間ステートチャートの提案

3.1 仕様記述と検証の要件

本稿では，家電製品などの小規模な組み込み型ソフトウェアのタイミング記述を含む仕様記述と検証を対象とする．その方法の要件として以下を考える．

①仕様が作成しやすく理解しやすいこと

プロセスを記述単位とした形式的仕様記述がよい．このことは，時相論理やペトリネットよりもオートマトンやモードチャート，プロセス代数などが適していることを示す．

②仕様が機械的に検証できること

形式的モデルに準拠して，検証アルゴリズムが簡単なものがよい．このことは，オートマトンや時相論理，ペトリネットなどが適していることを示す．

③システム仕様と検証仕様が統一的であること

一般的に，タイミング記述を含むシステム仕様は時間グラフ（時間オートマトン）で表現できる．検証仕様を時間オートマトンで記述すればよい．

以上より，時間オートマトンによるシステム仕様と検証仕様の統一的記述による検証が要件を満たすことがわかる．また，システム仕様記述の状態の組み合わせ爆発は，ステートチャート[10]により克服可能と考えられる．

我々は，以上の考察により次の手法を提案する．仕様記述手法としては，時間オートマトンによりステートチャートを形式化した時間ステートチャートを採用する．検証性質記述も同様な時間オートマトンを採用して，言語包合問題により検証する．

3.2 仕様記述言語

（1） 構 文 定 義

本構文定義は，時間 Buchi オートマトンによりステートチャートを形式化したものである．

［定義 1］（仕様記述言語の構文定義）

仕様記述言語は $\langle\Sigma, S, S0, C, E, F, \rho, \psi\rangle$ の 8 つ組で

定義できる．

ここで，各項は次のとおりである：

Σ：有限イベント集合；

S：有限状態集合；

S0：初期状態集合 ($S0 \subseteq S$)；

C：クロックの有限集合；

E：時間付き遷移関数；

$(E \subseteq 2^S \times 2^S \times \Sigma \times 2^C \times \Phi(C))$

ただし，

$\Phi(C)$：クロック Cのタイミング制約式 δ

$\delta := C<d \mid C<d \mid \neg\delta \mid \delta_1 \wedge \delta_2 \mid C := 0$

なお，C：クロック変数，d：クロック定数

F：受理状態集合 ($F \subseteq S$)；

ρ：$S \to 2^S$ であり各状態のサブ状態を決める階層関数である

ψ：$S \to \{AND, OR\}$ であり各状態の型を決める型関数である □

イベントにより，無限回遷移する状態と受理状態集合Fとの共通集合が空集合でないならば受理される．$\psi(S)=AND$ ならば，$\rho(S)$ はSの AND 分割（並行化）である．$\psi(S)=OR$ ならば，$\rho(S)$ はSの OR 分割（階層化）である．

また，時間ステートチャートの動作イメージを簡単に説明する．イベントeが状態Sに到着すると，次の①〜③の規則が状態階層の上位から下位に再帰的に適用される：

①$\psi(S)=AND$ ならば，S の全サブ状態 $\rho(S)$ にイベントeがブロードキャスト通信される

②$\psi(S)=OR$ ならば，どれか1つのサブ状態 $\rho(S)$ にイベントeが通信される

③$\psi(S) \neq AND$ かつ $\psi(S) \neq OR$ ならば，イベントeが遷移条件を満たせば状態Sは遷移する

（2） 意味定義

プロセス代数 (CSP 等) のトレース意味論[17]を拡張した時間トレース意味論により意味を定義する．時間トレース意味論の基本は，観測可能なイベント集合をプロセスと同一視して，イベントトレースによりプロセス動作を定義するものである．時間付きイベント列を扱うために，イベントとその生起する時間の順序対により意味を定義する．

［定義2］（仕様記述言語の意味定義）

仕様記述言語の意味はイベント σ_i が時間 τ_i で生起することを表示する順序対 (σ_i, τ_i) の遷移列で以下のように表現する．

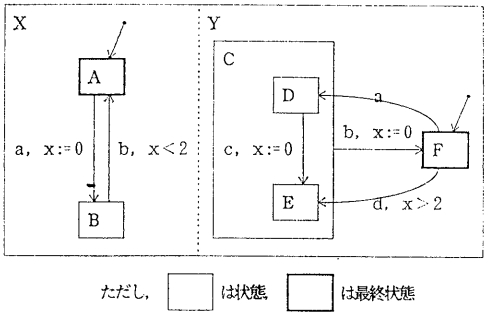


図 5 仕様記述言語の記述例
Fig. 5 Example of specification language.

$(\sigma_1, \tau_1) \to (\sigma_2, \tau_2) \to (\sigma_3, \tau_3) \to$

ただし，順序対 (σ_i, τ_i) は無限列であり，τ_i の領域は実数であり，いわゆる dense-time モデル（稠密時間モデル）の範疇に入る．また，時間 τ_i は以下の条件を満たす．

①初期条件：$\tau_0=0$ が開始時間である

②単調性：$\forall i \geqq 0, \tau_i < \tau_{i+1}$

③前進性：$\forall t \in R, \exists i, \tau_i > t$ □

（3） 仕様記述言語の記述例

仕様記述例を**図5**に示す．なお，A〜Fは状態，xはクロック変数，a〜dは入力アルファベット（イベント），x:=0 はリセット式，x<2 などはタイミング制約である．状態のXとYは AND 分割され，DとEはCの OR 分割である．状態Xでは，初期状態がAであり，aが入力されるとクロックxがリセットされて状態Bに遷移して，状態Bでbが入力されてクロック x<2 が満たされるとAに遷移する．状態Yも同様である．例えば，時間トレース列

(a, 2)→(b, 3.5)→(a, 4)

が入力されると状態のXとYは次のように遷移し，トレースにより動作的意味が定義できる．

$$X: \langle A, [0]\rangle \xrightarrow{(a,2)} \langle B, [0]\rangle \xrightarrow{(b,3.5)} \langle A, [1.5]\rangle \xrightarrow{(a,4)} \langle B, [0]\rangle$$

$$Y: \langle F, [0]\rangle \xrightarrow{(a,2)} \langle D, [2]\rangle \xrightarrow{(b,3.5)} \langle F, [0]\rangle \xrightarrow{(a,4)} \langle D, [0.5]\rangle$$

ただし，[x] はクロック値である．

なお，最終状態は，AおよびFであり，AおよびFが無限回繰り返される時間トレース列が受理される．

3.3 検証方式

3.3.1 検証性質

リアルタイムソフトウェアの検証性質としては次の安全性と活性が知られている[12]．

①安全性 (safety)

安全性は静的な性質であり，不変性としても知られており，「何も悪いことは起きない」ことである．つまり，ある過去の好ましい性質が満たされていることを意味する．具体的には，計算結果が正しいといった部分正当性やクリテカルセクションの相互排他などの大域的不変性などである．

②活性 (liveness)

活性は動的な性質であり，終局性や進行性としても知られており，「何か良いことが起こるだろう」ということである．つまり，ある有限の時点における性質がある数だけ成立することを意味する．具体的には，計算が停止し計算結果が正しいといった完全正当性や先に進めるプロセスが1つもないデッドロックフリー性，どのプロセスも公平に実行される公平性（飢えがない）などである．

3.3.2 検証の性質記述

検証性質記述の条件は，記述能力が高く，補集合演算の閉包性を有することである．この条件を満たす言語は，状態のベキ集合を受理する決定性時間 Muller オートマトンである．

[定義3]（検証記述言語の構文定義）

検証記述言語は〈Σ, S, S0, C, E, F〉の6つ組で定義できる．

ここで，各項は次のとおりである：

Σ：有限イベント集合；

S：有限状態集合；

S0：初期状態集合 (S0⊆S)；

C：クロックの有限集合；

E：時間付き遷移関数；

$(E \subseteq S \times S \times \Sigma \times 2^C \times \Phi(C))$

ただし，

$\Phi(C)$：クロック Cのタイミング制約式 σ

$\delta := C<d \mid d<C \mid \neg\delta \mid \delta_1 \wedge \delta_2 \mid C := 0$

なお，C：クロック変数，d：クロック定数

F：状態のベキ集合である受理状態集合

$(F \subseteq 2^S)$；

なお，$|S0|=1$ であり状態遷移は決定的である．□

3.3.3 検証アルゴリズム

検証の基本方針は，時間ステートチャートと決定性時間 Muller オートマトンの言語包合問題であり，次の手順から構成する．

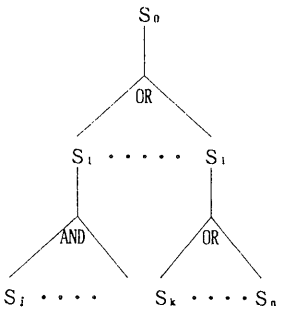
①時間ステートチャートから時間 Buchi オートマトンの積を生成すること

②積時間 Buchi オートマトンと決定性時間 Muller オートマトンの補集合から時間グラフ（積オートマトン）を生成すること

③時間グラフの閉ループがなければ，検証性質記述が満たされると判定すること

3.3.3.1 時間ステートチャートから時間 Buchi オートマトンの積の生成

基本的には，時間ステートチャートの AND 分割は状態の積を生成し，OR 分割は状態階層を平坦にする．この場合，状態遷移のクロック制約式やリセット



ここで，状態集合＝{S0, S1, ⋯, Sn}

S0：初期状態

AND, OR：時間ステートチャートの AND/OR 分解

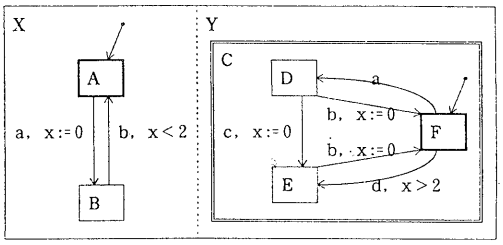
図6 時間ステートチャートの状態ツリー

Fig. 6 State tree of timed statechart.

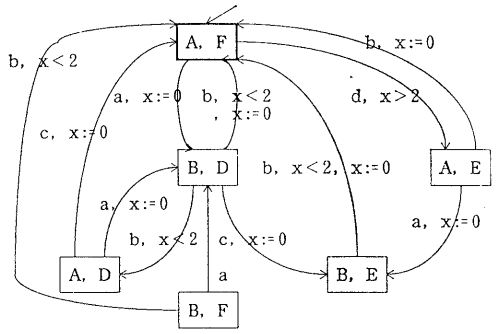
```
／＊ 時間ステートチャートの状態ツリーを生成する ＊／
while（根から葉まで横型探索順序で以下を繰り返す）
  {
    if（ρ（s）≠0かつφ（s）=OR）
      then
        ρ（s）のサブ状態集合をORツリーとして生成する
      else
        if（ρ（s）≠0かつφ（s）=AND）
          then
            ρ（s）のサブ状態集合をANDツリーとして生成する
  }

／＊ 状態ツリーに従って積のオートマトンを生成する ＊／
／＊ 生成順序は階層レベルであり，同一レベルは並行的に生成する ＊／
while（最下位階層の節点から根の状態まで以下を繰り返す）
  {
    if（生成順序が同一のツリーがあるか）
      then
        並行的に同一レベルの複数のツリーを処理する
          （サブルーチンANDORを並行的に呼び出す）
      else
        単一のツリーを処理する
          （サブルーチンANDORを呼び出す）
  }

／＊ ANDとORの処理をするサブルーチン ＊／
procedure ANDOR
{
  if（ρ（s）≠0かつφ（s）=OR）
    then
      ρ（s）をフラットにし，sを生成する
    else
      if（ρ（s）≠0かつφ（s）=AND）
        then
          ρ（s）のサブ状態集合の積を生成する
}
```

図7 積オートマトンの生成アルゴリズム

Fig. 7 Generation algorithm of product of automata.



（a） OR ツリーの展開例
（a） Example of OR tree



（b） AND ツリーの展開例
（b） Example of AND tree

図 8 図5の積オートマトン生成例
Fig. 8 Example of product of autumata in Fig. 5.

式は各遷移の論理積となり，クロック集合は和集合となる．また，積オートマトンの受理状態集合は，各オートマトンの状態への射影の組み合わせである．

具体的なアルゴリズムを以下に説明する．**図6**に示すように，時間ステートチャートは唯一の根の状態を持つ AND/OR ツリーである．積を生成する方法として，最下位層から，AND ツリーは積を，OR ツリーは状態を展開する．**図7**のアルゴリズムにより，**図8** の積オートマトンを生成する．

［定義4］（積時間 Buchi オートマトンの生成アルゴリズム）

生成アルゴリズムは次の計算手順である．

①生成順番の決定

まず，最上の根の状態から型関数や階層関数を適用しながら，図6のような状態図ツリーを作成して積オートマトンの作成順序を決定する．

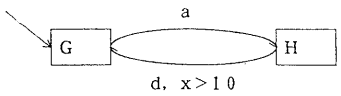


図 9 検証性質の記述例
Fig. 9 Example of verification property specification.

②積の生成

状態図ツリーに従って，最下層から根の状態まで，積を生成する．なお，同一階層は同時並行的に処理できる．図6では，S_j を含む AND ツリーと S_k を含む OR ツリーは同時並行的に処理できた．次に，S_i を含む OR ツリーを処理する． □

なお，OR ツリーの処理は階層状態をフラットにするだけなので自明である．AND ツリーの処理は概ね以下の手順で生成できる：

①AND のすべての状態の初期状態の組み合わせから積の初期状態を作る．

②初期状態からイベントで遷移可能な状態の組み合わせを生成する．ただし，タイミング制約は AND 結合する．

3.3.3.2 時間グラフの生成

時間グラフの生成は，上記の積生成と同様なので，省略する．ここでは，安全性として**図9**の性質記述を考えると，**図10**の時間グラフが生成できる．

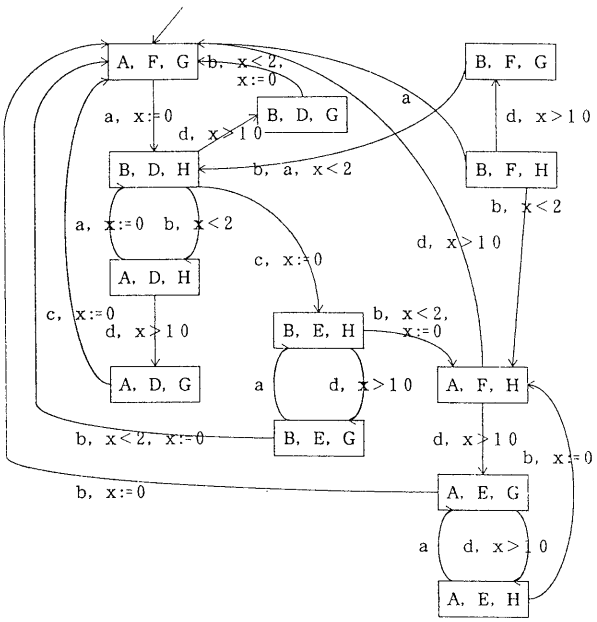


図 10 検証例
Fig. 10 Example of verification.

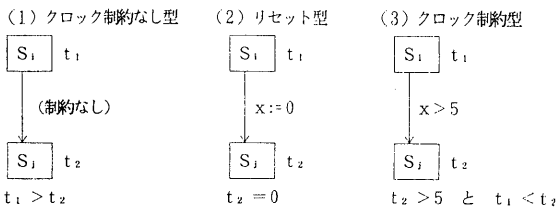


図 11 状態へのクロック値割当の基本パターン
Fig. 11 Pattern of assignment of clock-value to state.

3.3.3.3 閉ループの発見

時間グラフの閉ループを発見する．すなわち，時間 Buchi オートマトンと決定性時間 Muller オートマトンの補集合の積が時間付き言語を受理するかどうかを判定する．図8の積オートマトンは {A, F} が受理集合であり，図9の決定性時間 Muller オートマトンの補集合は {{G}, {H}, {}} が受理集合である．ゆえに，図10の受理集合は {A, F, G} と {A, F, H} である．しかし，図10の時間グラフは決定性時間 Muller オートマトンの補集合の受理条件を充足する閉ループは存在しないので，仕様は検証性質を充足する．以下にアルゴリズムを整理する．

[定義5] (閉ループ発見アルゴリズム)
閉ループ発見の計算手順は次のとおりである．

①状態へのクロック制約の割り付け

図11 のような基本パターンに従って，状態にクロック制約を割り付けて，(状態，クロック制約) の順序対からなるクロックなしの状態図を生成する．

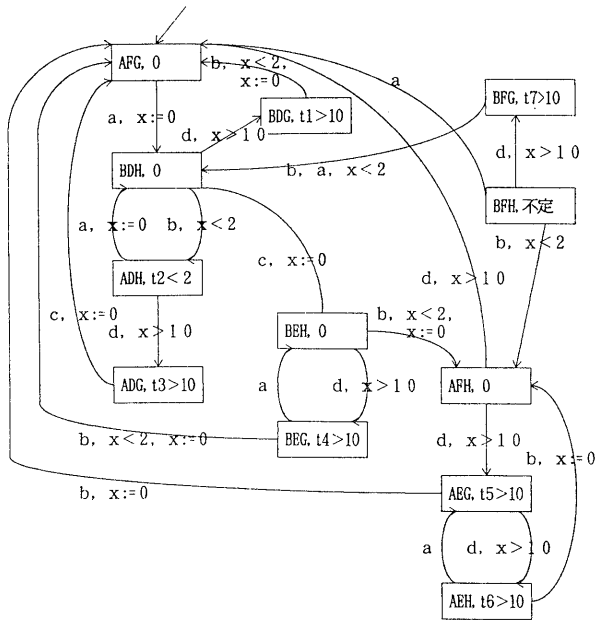
②クロック制約の判定

受理集合を含むループが存在するかどうかをクロック制約の連立不等式の無矛盾性の機械的な判定から行う． □

その詳細なアルゴリズムを**図12**に示す．**図13**に事例を示すように，クロック制約の連立不等式に矛盾がなければ (充足可能)，該当ループを満たす入力列は存在することになる．

```
if (閉ループは存在するか)
 then    /* 閉ループが存在する */
   ループの初期状態にクロック値0を割り当てる
   /* ループ内の全状態遷移をたどる                          */
   /* その時，遷移条件より状態にクロック制約式を割り当てる */
   while (ループ内の全状態遷移をたどる)
     {
       if (遷移条件にクロック制約がないか)
         then
           遷移先クロック値>遷移元クロック値 を制約式にする
       if (遷移条件にリセット式があるか)
         then
           遷移先クロック値=0 を制約式にする
       if (遷移条件にクロック制約があるか)
         then
           クロック制約と，遷移先クロック値>遷移元クロック値
           を制約式にする
     }
   /* すべてのクロック制約式の矛盾を導く */
   クロックの制約式の矛盾性を判定する
   if (クロック制約式に矛盾がないか)
     then
       受理言語は空集合でない
     else
       受理言語は空集合である
else    /* 閉ループが存在しない */
   受理言語は空集合である
```

図 12 検証のアルゴリズム
Fig. 12 Algorithm of verification.



例えば，

ループ AFG→BDH→ADH→ADG→AFG→… の制約式は

t2< 2 (ADH)……①

t3>10 (ADG)……②

t2<t3 (ADG)……③

である．

①と②より t2<t3 であり，③と矛盾しないので，このループは可能である．

なお，t1, …, t7 はクロック変数である．

図 13 クロック割当の状態遷移図の例
Fig. 13 Example of clock assigned state transition diagram.

4. 例題による仕様記述と検証の説明

本章では，簡単化した時計問題をもとに，提案した仕様記述と検証の有効性を以下に示す．

(1) 問題の概要

本問題は，**図14**に示すような構成である．時計は制御部と表示機能，beep 機能からなり，制御部は表示機能と beep 機能に制御信号をだ

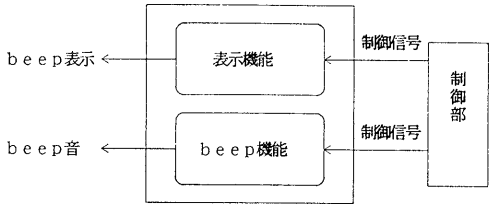


図 14 時計問題の説明図
Fig. 14 Diagram about watch problem.

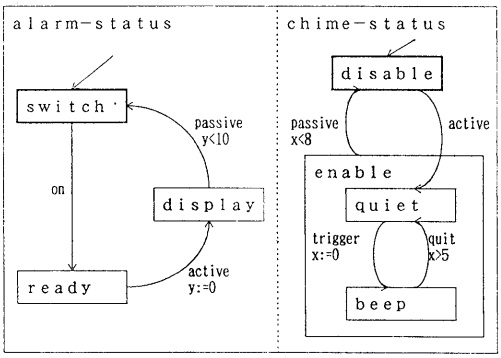


図 15 時計問題の時間ステートチャート
Fig. 15 Timed statechart of watch problem.

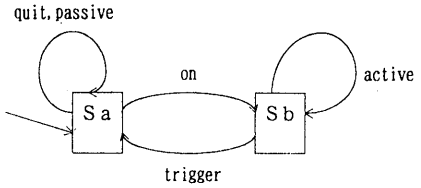
しながら，これらを制御する．今回の仕様記述の対象は，表示機能と beep 機能とする．時計は，表示機能をつかさどる alarm-status と beep を鳴らす chime-status から構成されるとする．これらの2つは，相互作用する並行プロセスであり，各々のプロセスを時間オートマトンでモデル化できる．

この場合，静的な性質としては，alarm-status が作動可能状態になってから初めて beep が鳴りだすこと，動的な性質としては，beep が一度動作すると，次回に beep が表示可能となるために，必ず8時刻以上が必要であるなどが保証されるべきであるとする．

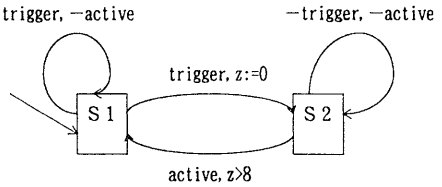
（2） 仕様記述と性質記述の例

上記 alarm-status と chime-status の時間ステートチャートを**図 15** に示す．2つの構成要素を並列動作するので，AND 分解で表現できて，chime-status はサブ状態が存在するので，OR 分解できる．最終状態は switch と disable であり，これらの無限の繰り返しが受理される．このように，時間ステートチャートの仕様記述では，従来のオートマトンでは表現できない並列性や階層性がうまく表現できて，理解しやすく改良しやすい．

また，充足すべき性質の記述は次の2つが考えられる．ひとつは作動可能状態になってから初めて beep

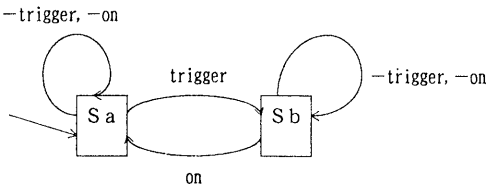


alarm-status が on になってから初めて beep が鳴る
（a） 安全性の記述
（a） Specification of safety

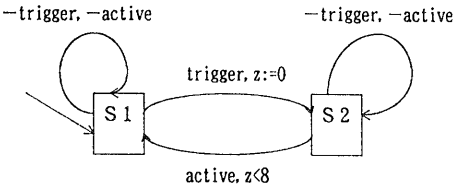


beep が一度動作すると，次回に beep が表示可能となるために，必ず8時刻以上が必要である
（b） 活性の記述
（b） Specification of liveness

図 16 時計問題の正当な性質記述の例
Fig. 16 Example of valid property specification.



alarm-status が on になるまえに beep が鳴る
（a） 安全性の記述
（a） Specification of safety



beep が一度動作すると，次回に beep が表示可能となるためには，8時刻以下でよい
（b） 活性の記述
（b） Specification of liveness

図 17 時計問題の不当な性質記述の例
Fig. 17 Example of invalid property specification.

が鳴りだすことであり安全性 (safety) である．**図 16**（a）に記述例を示す．他方は次回の beep の表示は8時刻以上が必要であることであり，活性 (liveness) である．図 16（b）に記述例を示す．なお，-active は active 以外のイベントを表す．両方の性質記述も Sa

と Sb または S1 と S2 の無限の繰り返しが受理される．また，逆に，不当な性質記述の例を**図 17** に示し，検証例で取り上げる．

（3） 性質の検証例

本稿のアルゴリズムを実際にインプリメントして検証を行ったので，以下に結果を説明する．

まず，時間ステートチャートから積オートマトンを生成して，図 16(a)の検証性質を検証する場合を考える．この場合の時間グラフは**図 18** である．受理集合は {{Sa}, {Sb}, {}} と {switch, disable} の積の状態が無限に繰り返されるループが存在するかどうかである．この場合は閉ループが存在しないので，性質は充足される．(b)も同様な手続きで充足が検証できる．また，図 17 の(a)と(b)の両方ともに受理集合が含むループが存在して，これらの性質は充足されないことが容易に示せる．

図 17 では，(a)に，

{switch, disable, Sa} →
{switch, disable, Sb} →
{ready, disable, Sa} →
{display, quiet, Sa} →
{switch, disable, Sa} →

が存在し，(b)に，

{switch, disable, S1} → {ready, disable, S1} →
{display, quiet, S1} → {display, beep, S2} →
{switch, disable, S2} → {switch, disable, S1} →

が存在する．

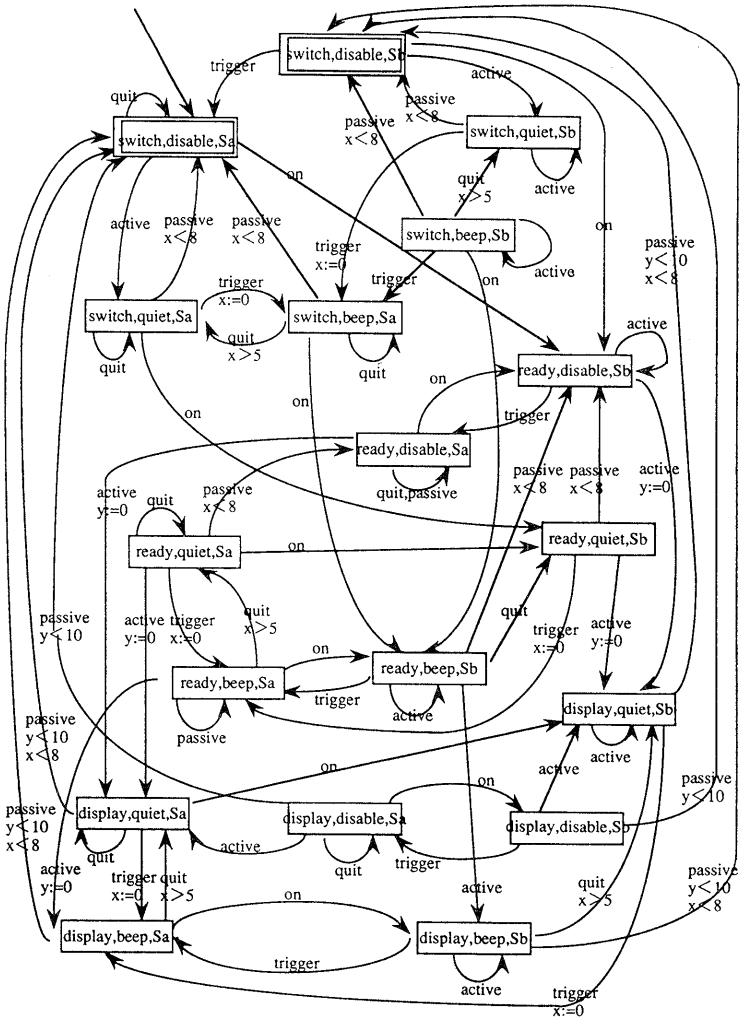


図 18 時間グラフ
Fig. 18 Timed graph.

5. ま と め

本稿では，時間ステートチャートに基づく仕様記述と検証の手法を提案し，例題を用いてその有効性を確認した．本手法の主要な特徴は次のとおりである：

①ステートチャートの拡張により形式的かつ構造的な仕様記述を可能とした

②システム仕様と検証仕様を統一的に記述可能とした

③言語包含問題によりタイミング検証を可能とした

本手法には今後，以下の課題が残っている．

①ソフトウェアの規模が大きくなるにつれて検証コストが大きくなる．積オートマトンの生成や検証のアルゴリズムでは，状態数の多項式オーダの時間がかかる．

②動作と状態に注目したが，やや抽象度が低い．そこで，より抽象度の高い仕様から時間オートマトンを自動生成する必要がある．

現在，これらを解決すべく以下を検討している．

①積生成や検証の状態数を縮小するために，検証の局所化や BDD (Binary Decision Diagram) の適用
②より抽象度の高い構成要素に着目するために，時相論理やプロセス代数による動作と状態の統一的な記述からの時間オートマトンの自動生成

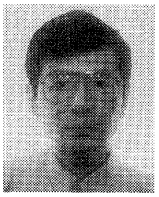
なお，現在，作成したアルゴリズムの詳細な評価と改善を行っている．

謝辞 日頃より本研究を御支援いただいた島根大学理学部情報科学科西村正太郎教授に感謝します．

参考文献

1) Kavi, K. M.: Real-time Systems, Abstraction, Language, and Design Methodologies, IEEE CS (1992).
2) Hatley, D. J. and Pirbhai, I. A.: *Strategies for Real-Time System Specification*, Dorset House (1988).
3) Jahanian, F. and Stuart, D. A.: A Method for Verifying Properties of Modechart Specification, *Proc. Real-Time Systems Symposium*, pp. 12-21 (1988).
4) Berthomieu, B. and Diaz, M.: Modeling and Verification of Time Dependent Systems Using Time Petri Nets, *IEEE Trans. Softw. Eng.*, Vol. 17, No. 3, pp. 259-273 (1991).
5) Milner, R.: *Communication and Concurrency*, Prentice Hall (1989).
6) Sifakis, J.: Automatic Verification Methods for Finite State Systems, *LNCS (Lecture Notes in Computer Science) 407*, p. 382 (1989).
7) Bakker, J. W., de Roever, W. P. and Rozenberg, G.: Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, *LNCS 354*, p. 713 (1988).
8) Hopcroft, J. E. and Ullman, J. D.: *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley (1979).
9) Coleman, D., Hayes, F. and Bear, S.: Introducing Objectcharts or How to Use Statecharts in Object-Oriented Design, *IEEE Trans. Softw. Eng.*, Vol. 18, No. 1, pp. 9-18 (1992).
10) Harel, D.: Statecharts: A Visual Formalism for Complex Systems, *Science of Computer Programming*, Vol. 8, pp. 231-274 (1987).
11) Jones, C. B.: *Systematic Software Development Using VDM 2th*, p. 333, Prentice Hall (1990).
12) Leeuwen, J. V.: Formal Models and Semantics, *Handbook of Theoretical Computer Science Vol. B*, pp. 997-1072 (1990).
13) Alur, R. and Dill, D.: The Theory of Timed Automata, *LNCS 600*, pp. 45-73 (1992).
14) Alur, R. and Dill, D.: Automata for Modeling Real-Time Systems, *LNCS 443*, pp. 322-335 (1990).
15) 小林孝次郎，高橋正子：オートマトンの理論，pp. 109-130, 共立出版 (1983).
16) Eilenberg, S.: *Automata, Language, and Machines, Vol. A*, pp. 358-393, Academic Press (1974).
17) Hoare, C. A. R.: *Communicating Sequential Processes*, pp. 256, Prentice-Hall (1985).

(平成6年4月15日受付)
(平成6年9月 6 日採録)



山根 智（正会員）

山口県生まれ．1984 年京都大学工学研究科修士課程修了．同年富士通(株)入社．通信ソフトウェアの研究開発に従事．その後，島根大学理学部情報科学科勤務，現在に至る．通信ソフトウェアなどのリアクティブシステムの形式的な仕様記述や検証，仕様自動生成に関する研究に従事．形式言語理論や論理学，プロセス代数，人工知能などに興味を持つ．EATCS, IEEE Computer Society, ACM, 人工知能学会等各会員．