ハードリアルタイムシステムの形式的仕様記述と形式的検証

山根 智†

ハードリアルタイムシステムは外部環境との相互作用があり，タイミング制約が厳しく，並行プロセスにより構成される．このために，仕様が正しいことを保証することが重要である．本論文では，時間ステートチャートを一般化して，並行性やタイミング制約，機能の統合的な仕様記述を実現する．さらに，Pnueli らのクロック遷移システムにより，時間ステートチャートの操作的意味を表現する．それを基にして，演繹的証明により，時間ステートチャートが安全性や活性を充足することが検証できることを示す．最後に，クロック遷移システム上で詳細化検証の公理系を開発して，時間ステートチャートの詳細化の検証を示す．

Formal Specification and Verification Method for Hard Real-time Systems

Satoshi Yamane†

Hard real-time systems have ongoing interactions with external environments, and consist of concurrent processes. It is important to verify whether hard real-time systems satisfy safety and liveness properties or not. In this paper, we integrally specify concurrency, timing constraints, functions by generalizing our timed statechart. Moreover, we represent operational semantics of timed statechart using Pnueli's clocked transition systems. Next, we verify whether timed statechart satisfies safety and liveness properties or not by deductive proofs. Finally, we propose a refinement verification rule and verify whether concrete specification refines abstract specification or not.

1. はじめに

通信システムや制御システムの多くはハードリアルタイムシステムである[1)]．さらに，これらは他のシステムと協調して動作するリアクティブシステムであることもしばしばである．したがって，これらのシステムの設計においては，各システムにおける処理時間が他のシステムに及ぼす影響や，システム間の通信遅延といった実時間性をも考慮する必要がある．このことにより,(リアクティブ) ハードリアルタイムシステムの設計作業は複雑になりがちであり，また，システムの信頼性保証も一般的には困難である．したがって，ハードリアルタイムシステムの信頼性保証ができることは重要である．本論文では，ハードリアルタイムシステムの信頼性保証のために，形式的仕様記述言語と形式的検証手法を提案する．具体的には，以下のとおりである：まず，時間ステートチャート[2)]を一般化して，並行性やタイミング制約，機能を統合的かつ形式的に仕様記述できる言語を提案する．従来の形式的仕様記述言語では，並行性やタイミング制約，機能が統合的かつ形式的に仕様記述できない．さらに，一般化した時間ステートチャートが安全性や活性といった望ましい性質を充足するかどうかを形式的に演繹的検証する手法を提案する．そして，一般化した時間ステートチャートにより，ハードリアルタイムシステムを階層的に詳細化して設計できる手法，すなわち，詳細化を形式的に演繹的検証する手法を提案する．従来の形式的検証手法では，ハードリアルタイムシステムの形式的な詳細化演繹的検証手法は存在しない．

従来のハードリアルタイムシステムの形式的手法の主要な研究には，以下のようなものがある：

(1) Alur らは時間オートマトンを開発して，ハードリアルタイムシステムの形式的仕様記述言語とした[3)]．さらに，時間オートマトンの形式的検証手法を開発した[4)]．しかし，時間オートマトンは有限状態遷移システムであり，複雑なシステムが仕様記述できない．

(2) Lynch らは，時間 I/O オートマトンを開発して，無限な状態集合を有するシステムの仕様記

† 鹿児島大学工学部情報工学科
Department of Information and Computer Science, Kagoshima University

述と詳細化の検証の公理系を開発した[5]．しかし，時間 I/O オートマトンは抽象的であり，動作モデル（いわゆる状態遷移モデル）や機能モデル（いわゆるデータ処理）などの区別が明確でないために，実際のシステムの設計に使えない．

(3) Kesten らは，クロック遷移システム（clocked transition systems）を開発して，無限な状態集合を有するシステムの仕様記述と安全性や活性の検証の公理系を開発した[6]．しかし，クロック遷移システムは並行性や機能などが陽に表現できないために，仕様記述言語には適さない．さらに，クロック遷移システムには，詳細化の検証の公理系が開発されていない．

本論文では，まず，時間ステートチャート[2]を一般化して，並行性やタイミング制約，機能を統合的に仕様記述する．なぜならば，既存の時間ステートチャート[2]では，状態遷移モデルをタイミング制約記述で拡張したのみであり，値の変数への代入などの機能が表現できないので，仕様記述言語としては表現能力が弱いからである．

次に，演繹的証明により，時間ステートチャートの安全性や活性，詳細化を形式的に検証する．従来のステートチャートの研究では，動作モデルや機能モデルが独立しているために形式化が不十分であったり[7]，タイミング制約と機能を統合的に形式化していなかったりする[8]．また，ハードリアルタイムシステムの詳細化の検証の公理系が開発されていない[6]．

本論文では，まず，時間ステートチャートを定義して，その操作的意味をクロック遷移システム上で定義する．次に，安全性や活性を時相論理式で仕様記述して，クロック遷移システムが時相論理式を充足するかどうかを，演繹的証明で検証する手法を示す．次に，詳細化の検証の公理系により，時間ステートチャートの詳細化を検証する．

本論文の構成は，次のとおりである．2 章では，時間ステートチャートを定義する．3 章では，時間ステートチャートの操作的意味をクロック遷移システム上で定義して，安全性や活性の検証手法を定義する．4 章では，時間ステートチャートの操作的意味をクロック遷移システム上で定義して，詳細化の検証手法を定義する．最後に，5 章では，まとめと今後の課題を述べる．

2. 時間ステートチャートによる仕様記述

一般的には，ハードリアルタイムシステムは動作モデルと機能モデルから構成される．本論文では，ハードリアルタイムシステムは動作モデルが大きく，機能モデルは小さいと考える．これにより，動作モデルのアクションの中に機能モデルを押し込む．そして，タイミング制約が記述できるように拡張して，時間ステートチャート[2]が並行性と機能モデル，タイミング制約を統合的に表現できるようにする．

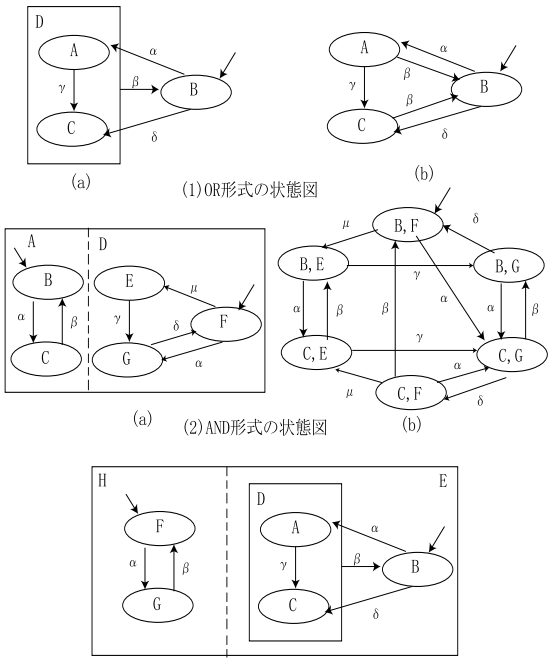

図 1 ステートチャートの例
Fig. 1 Example of statechart.

2.1 ステートチャートの概要

ステートチャート[9]はシステムが時間的に反応しているときに，動作がいつ，どのように，なぜ発生しているかを表現する．状態組合せ爆発を抑制するために，ステートチャートでは，AND 形式（並列形式）と OR 形式（階層形式）の状態遷移図で表現して，イベントがブロードキャスト通信される機構が備わっている．ブロードキャスト通信とは，ステートチャートを構成するすべての AND 形式と OR 形式の状態遷移図にイベントが通信されることを意味する．さらに，状態への遷移および状態からの遷移がどの階層でも実現できるようになっている．**図 1** に従って，ステートチャートの基本概念を簡単に説明する．図 1 (1) (a) の OR 形式の状態遷移図では，状態 D に存在することは状態 A または状態 C に存在することである．ゆえに，図 1 (1) (a) は図 1 (1) (b) と等しい．また，図 1 (2) (a) の AND 形式の状態遷移図では，状態 A と状態 D が並行動作して，イベント α とイベント β は状態 A と状態 D にブロードキャスト通信される．ゆえに，図

1 (2) (a) は図 1 (2) (b) と等しい．以上の OR 形式と AND 形式を組み合わせると，図 1 (3) のような一般的なステートチャートの基本形が構成できる．

2.2 時間ステートチャート

本論文で対象とする時間ステートチャートは，ステートチャートを拡張したものであり，AND/OR 形式の状態遷移図とブロードキャスト通信機構があり，状態遷移は δ，λ，e/a または e/a でラベル付けされているとする．なお，e が外部からのイベントの場合は δ，λ，e/a であり，e が内部イベントの場合は e/a である．ここで，δ はタイミング制約式，λ はリセット式，e はイベント，a はアクションとする．

(1) タイミング制約式 δ は，クロック集合 C ($x \in C$) と非負整数の時刻定数 d により以下のように帰納的に定義される：
  (a) $x \leq d$ や $d \leq x$ は δ である．
  (b) δ_1 と δ_2 が δ ならば，$\neg\delta_1$ や $\delta_1 \wedge \delta_2$ は δ である．

(2) λ はリセット式の集合であり，リセット式は $x := 0 (x \in C)$ で表現する．

(3) イベントはある変数が特定の値に等しいといった条件文である．

(4) アクションはある変数をある値に代入するといった実行文である．

さらに，以下の前提条件を仮定する．

(1) 同期性仮定
  システムは環境よりも無限に速く動作して，環境からの刺激に対してシステムは瞬時に反応すると仮定する．たとえば，$t_0 : a/e_1, t_1 : e_1/e_2, \ldots, t_n : e_n/b$ の状況において，a と b は環境との刺激・応答として，$e_1, e_2, \ldots, e_n$ は内部イベントまたは内部アクションとすると，$t_0, t_1, \ldots, t_n$ は瞬時にチェイン状に発生する．ここで，各 $t_i (i \geq 0)$ はステップと呼び，$\bigcup t_i$ はスーパーステップと呼ぶ．この仮定は，モデル化を単純にするための抽象化である．

(2) 因果関係保存
  ステップに順番を付与して，順番が大きな遷移から小さな遷移が発生しないようにする．たとえば，$t_1 : a/b$，$t_2 : b/a$ の状況において，$t_1, t_2, t_1, \ldots$ は無限に発生する可能性があるが，因果関係の保存により，t_1，t_2 で遷移が停止する．この仮定は，内部イベント/アクションの無限動作を回避するために必要である．

(3) プライオリティの表現
  同時に発生したイベントのどれが実行すべきかを指定できるようにする．たとえば，$t_1 : a/$，$t_2 : b/$ の状況において，a と b が同時に発生すると，非決定的に，どちらかが動作する．しかし，リアルタイムシステムでは，優先度を付けて実行させたいことがある．そこで，否定表現により，優先度を指定する．この例では，$t_1 : a \cdot \overline{b}/$，$t_2 : b/$ のように記述することにより，a と b が同時に発生したときに，$t_2 : b/$ が動作するように指定する．これにより，実行の優先度が指定できる．

次に，時間ステートチャートを形式的に定義する．

Definition 1（時間ステートチャート）
時間ステートチャートは，$\mathbf{TSC} = (Event, Act, C, Q, Q_0, E, \rho, \phi)$ の 8 つ組で定義される．

(1) $Event$ はイベントの有限集合．
(2) Act はアクションの有限集合．
(3) C はクロック変数の有限集合．
(4) Q は有限状態集合．
(5) $Q_0 \subseteq Q$ は初期状態集合．
(6) $E \subseteq 2^Q \times 2^Q \times 2^C \times \Phi(C) \times 2^{Event} \times 2^{Act}$ は状態遷移関係．
  ただし，$\Phi(C)$ はクロック集合 C のタイミング制約式 δ であり，2^C はリセットされるクロック集合の集合である．
(7) $\rho : Q \to 2^Q$ は各状態の下位階層の状態集合を決める階層関数．
(8) $\phi : Q \to \{BASIC, OR, AND\}$ は各状態の型を決める型関数．

■

状態遷移関係 E は，現在の状態と次状態，リセットされるクロック変数，イベント，アクションのベキ集合とタイミング制約式のカルテジアン積の部分集合である．イベントは，ある変数が特定の値であるといった条件式である．ここで，変数を var_1，$var_2 \in VAR$，変数の値を $value_1$，$value_2 \in VALUE$ とすると，イベントは以下のように帰納的に定義される：

(1) $var_1 = value_1$ はイベント e である．
(2) e_1 と e_2 がイベント e ならば，$\neg e_1$ や $e_1 \wedge e_2$ はイベントである．

アクションは，ある変数へのデータの代入である．アクションは以下のように帰納的に定義される：

(1) $var_1 := value_1$ はアクション a である．
(2) a_1 と a_2 がアクション a ならば，$a_1 \parallel a_2$ はアクションである．ただし，$\parallel$ は並列動作することを意味する．

AND 形式により，並列関係にある状態集合が表現で

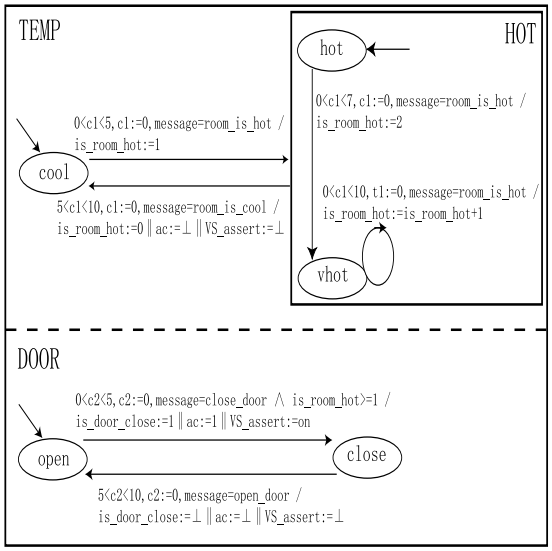

図 2 時間ステートチャートの仕様記述例
Fig. 2 Example of specification of timed statechart.

きる．OR 形式により，状態集合が階層構造を形成しており，階層関数 ρ によって下位階層の状態集合を定義する．また，各状態は，型関数 ϕ により，階層の一番上のルート状態であるか，または AND/OR 形式に属するかが定義される．

Example 1（時間ステートチャートによる仕様記述例）

簡単なエアコン制御システムを考える．エアコン制御システムは，センサ感知プログラムより送信される，温度計とドアに関するメッセージに反応して，アクチュエータ駆動プログラムにメッセージを送信してエアコンを作動する．すなわち，部屋が *hot* または *vhot* の状態でドアが *close* の状態のときには，アクチュエータ駆動プログラムにメッセージを送信してエアコンを作動して（$VS\_assert := on$ かつ $ac := 1$），そうでないときにはエアコンを停止させる（$VS\_assert :=\perp$ かつ $ac :=\perp$）．つまり，エアコン制御システムは，温度とドアの状態により，エアコン駆動を制御する．

このエアコン制御システムの要求仕様を，時間ステートチャートで仕様記述する．時間ステートチャートの状態空間は AND 形式（$TEMP$ と $DOOR$）と OR 形式（*hot* と *vhot*）から構成される．ここで，*message* は列挙型であり，$room\_is\_hot$ や $room\_is\_cool$，$close\_door$，$open\_door$ といった値をとる．また，is_room_hot や ac，is_door_close は $\perp$ または 0 を含む自然数であり，VS_assert はブール型であり，c_1, c_2 はクロック変数である．**図 2** にエアコン制御システムの時間ステートチャートによる仕様記述例を示す．その動作の概要は以下のとおりである：まず，動作の開始時は状態 *cool* と状態 *open* にあり，動作の開始時から 5 時刻未満にメッセージ $message = room\_is\_hot$ が送信されると，アクション $is\_room\_hot := 1$ を実行して状態 *hot* と状態 *open* に遷移する．次に，状態 *hot* と状態 *open* において，状態 *hot* に存在してから 7 時刻未満にメッセージ $message = room\_is\_hot$ が送信されると，アクション $is\_room\_hot := 2$ を実行して状態 *vhot* と状態 *open* に遷移する． ■

3. 時間ステートチャートの意味

本章では，クロック遷移システム上で時間ステートチャートの操作的意味を表現する．最初にクロック遷移システムを定義して，次に時間ステートチャートからクロック遷移システムへの変換方法を定義して，その操作的意味を定義する．

3.1 クロック遷移システム

ハードリアルタイムシステムの計算モデルとしてクロック遷移システム[6] を定義する．

まず，システム変数の有限集合を考える．システム変数は整数や実数などの型を持つ．我々はシステム変数にその型の値を割り付ける解釈として状態 s を定義する．すべての状態の集合を Σ とする．

Definition 2（クロック遷移システム）
クロック遷移システムは，$\mathbf{CTS} = (V, \Theta, \mathcal{T})$ の 3 つ組で定義される．ここで，

(1) V：システム変数の有限集合．集合 $V = D \cup C$ は，離散変数の集合 $D = \{u_1, \ldots, u_n\}$ とクロック変数の集合 $C = \{c_1, \ldots, c_k\}$ に分類できる．離散変数は任意の型がありえるが，クロック変数は実数型である．さらに，リセットされないマスタクロック $T \in C$ を導入する．

(2) Θ：初期条件．これは，すべての初期状態を特徴付ける表明である．$\Theta \rightarrow c_1 = \ldots = c_k = T = 0$ が要求される．

(3) $\mathcal{T}$：状態遷移の有限集合．各状態遷移 $\tau \in \mathcal{T}$ は関数 $\tau : \Sigma \rightarrow 2^{\Sigma}$ であり，各状態 $s \in \Sigma$ に次状態 τ-successor $\tau(s) \subseteq \Sigma$ を写像する．状態遷移 τ に関連する関数は表明 $\rho_\tau(V, V\prime)$ により表現される．$\rho_\tau(V, V\prime)$ は状態遷移関係と呼び，状態 $s \in \Sigma$ を τ-successor $s\prime \in \tau(s)$ に関係付ける．ただし，状態 s は V の型整合解釈であり，各変数 $v \in V$ に値 $s[v]$ を割り付ける．なお，システム変数の値は s の中の値や $s\prime$ の中の値として参照する．任意の $\tau \in \mathcal{T}$ に対して，$\rho_\tau \rightarrow T\prime = T$ が要求される．なお，

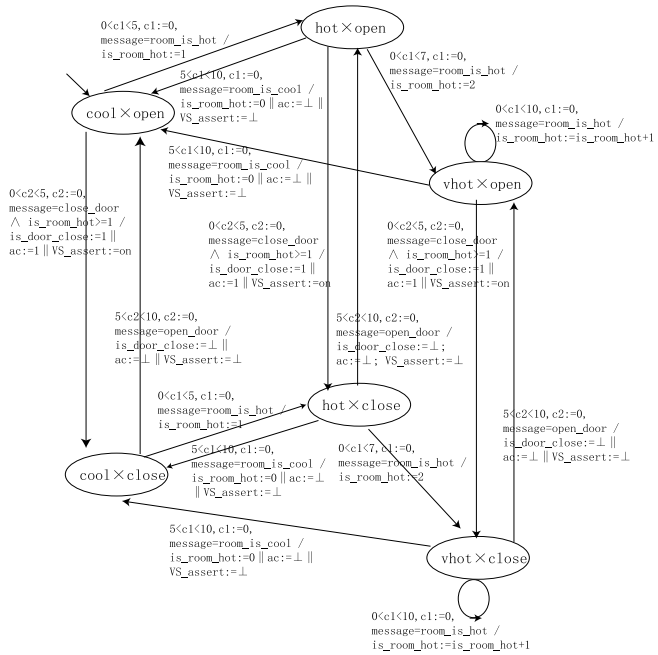

図 3 フラットな時間ステートチャートの構成例
Fig. 3 Example of construction of flat timed statechart.

$\rho_\tau \rightarrow T\prime = T$ は状態遷移が瞬時に発生することを意味する．

■

3.2 時間ステートチャートからクロック遷移システムへの変換

最初に時間ステートチャートからフラットな時間ステートチャートに変換して，次にフラットな時間ステートチャートからクロック遷移システムに変換する．

3.2.1 時間ステートチャートからフラットな時間ステートチャートへの変換

時間ステートチャートの状態集合は，唯一のルート状態を持つ AND/OR ツリーを構成している．ゆえに，文献 2) の手法により，ルート状態から，$\phi(q)$=AND ならば $\rho(q)$ の並列合成を構成して，$\phi(q)$=OR ならば $\rho(q)$ の階層を展開すれば，AND/OR 状態が存在しないフラットな時間ステートチャート $TSC\prime = (Event, Act, L, Q\prime, Q_0\prime, E\prime)$ を構成できる．ただし，$q \in Q$ とする．AND 形式の状態図を並列合成してフラットな状態図を構成すると，状態数が指数的に増大する．これは，積オートマトンの非空性判定問題が PSPACE 完全[10] であることに裏付けられており，避けることができない問題である．

時間ステートチャートから構成されるフラットな時間ステートチャートは，以下のように定義される．

Definition 3（フラットな時間ステートチャート） フラットな時間ステートチャートは $TSC\prime = (Event, Act, C, Q\prime, Q_0\prime, E\prime)$ の 6 つ組で定義される．

(1) $Event$ はイベントの有限集合．
(2) Act はアクションの有限集合．
(3) C はクロック変数の有限集合．
(4) $Q\prime = Q_1 \times Q_2 \times \ldots \times Q_n$（$n \in$ 自然数）は有限状態集合．ただし，n は並列動作する状態数である．
(5) $Q_0\prime \subseteq Q\prime$ は初期状態集合．
(6) $E\prime \subseteq Q\prime \times Q\prime \times 2^C \times \Phi(C) \times 2^{Event} \times 2^{Act}$ は状態遷移関係．

■

Example 2（時間ステートチャートによる仕様記述例） 図 2 のエアコン制御システムの時間ステートチャートから，フラットな時間ステートチャートが構成できる．**図 3** に，エアコン制御システムの時間ステートチャートから構成したフラットな時間ステートチャートを示す．

■

3.2.2 フラットな時間ステートチャートからクロック遷移システムへの変換

フラットな時間ステートチャートからクロック遷移システムへの変換方法を定義する．

Definition 4（クロック遷移システムへの変換） フラットな時間ステートチャート $TSC\prime = (Event, Act,$

$C, Q\prime, Q_0\prime, E\prime$) とクロック遷移システム **CTS** $= (V, \Theta, \mathcal{T})$ が与えられたとき，以下のように，フラットな時間ステートチャートからクロック遷移システムを構成する．

(1) $V = D \cup C$ は，離散変数の集合 $D = \{u_1, \ldots, u_m\}$ とクロック変数の集合 $C = \{c_1, \ldots, c_n, T\}$ からなる．D と C は以下のように構成される：
  (a) 離散変数の集合 D は並列動作する状態の集合 $Q\prime$ およびイベントとアクションに関わる変数の集合 VAR から構成される．$D = \{u_1, \ldots, u_n, var_1, \ldots, var_k\}$, ただし，$u_1 = q_1 \in Q_1, \ldots, u_n = q_n \in Q_n, var_1, \ldots, var_k \in VAR$. なお, n は並列動作する状態の数であり，k は変数の数であり，$m = n + k$ である．
  (b) C は時間ステートチャート上に現れるクロック変数 $c_1, \ldots, c_n$ およびマスタクロック T から構成される．

(2) Θ は初期条件であり，以下のように構成される：
  (a) D のすべての要素が初期値と等しくなるようにする．
  (b) C に関しては $c_1 = \ldots = c_n = T = 0$ であるようにする．

(3) $\mathcal{T}$ は状態遷移の有限集合である．状態遷移 $\tau \in \mathcal{T}$ に関連する関数を表明 $\rho_\tau(V, V\prime)$ により表現する．$\rho_\tau(V, V\prime)$ は現在の変数の値と次状態の変数の値との関係により定義する．

■

次に，フラットな時間ステートチャートの動作がどのようにクロック遷移システムの動作に反映されるかを定義することによって，時間ステートチャートの操作的意味を定義する．

Definition 5（時間ステートチャートの操作的意味）
フラットな時間ステートチャートの動作がどのようにクロック遷移システムの動作に反映されるかを定義することによって，時間ステートチャートの操作的意味を定義する．フラットな時間ステートチャート $TSC\prime = (Event, Act, C, Q\prime, Q_0\prime, E\prime)$ とクロック遷移システム **CTS** $= (V, \Theta, \mathcal{T})$ が与えられたとき，以下のように，操作的意味を定義する．
まず，フラットな時間ステートチャート $TSC\prime$ の動作を以下のように定義する．状態遷移は δ, λ, e/a または e/a でラベル付けされているので，以下の 2 つの場合を考える．

(1) δ, λ, e/a のとき

$$q_i\prime \xrightarrow{\delta, \lambda, e/a} q_{i+1}\prime$$

ただし，$q_i\prime, q_{i+1}\prime \in Q\prime$, δ はタイミング制約式，λ はリセット式，e はイベント，a はアクションである．

(2) e/a のとき

$$q_i\prime \xrightarrow{e/a} q_{i+1}\prime$$

ただし，$q_i\prime, q_{i+1}\prime \in Q\prime$, e はイベント，a はアクションである．

次に，上記に対応するクロック遷移システム **CTS** $= (V, \Theta, \mathcal{T})$ の動作を以下のように定義する：

(1) δ, λ, e/a のとき
$(u_1 = q_1 \wedge \ldots \wedge u_n = q_n \wedge var_1 = value_1 \wedge \ldots \wedge var_k = value_k \wedge t_1 < c_1 < t_1\prime \wedge \ldots \wedge t_n < c_n < t_n\prime \wedge t_T < T < t_T\prime) \longrightarrow (u_1 = {q_1}^{after} \wedge \ldots \wedge u_n = {q_n}^{after} \wedge var_1 = {value_1}^{after} \wedge \ldots \wedge var_k = {value_k}^{after} \wedge {t_1}^{after} < c_1 < {t_1\prime}^{after} \wedge \ldots \wedge {t_n}^{after} < c_n < {t_n\prime}^{after} \wedge {t_T}^{after} < T < {t_T\prime}^{after})$. なお，$after$ は遷移後の状態名や変数値などを意味する記号である．ただし，以下を満たす：
  (a) $q_i\prime = q_1 \times \ldots \times q_n$ および $q_{i+1}\prime = {q_1}^{after} \times \ldots \times {q_n}^{after}$.
  (b) δ が表現する時間領域と $t_1 < c_1 < t_1\prime \wedge \ldots \wedge t_n < c_n < t_n\prime$ が表現する時間領域との交わり[11] が空集合でない．
  (c) λ によりリセットされるクロック変数のリセット値と ${t_1}^{after} < c_1 < {t_1\prime}^{after} \wedge \ldots \wedge {t_n}^{after} < c_n < {t_n\prime}^{after}$ が表現する時間領域との交わり[11] が空集合でない．
  (d) イベント e による，ある変数がある値であるといった条件 $var_i = value_i$ は $var_1 = value_1 \wedge \ldots \wedge var_i = value_i \wedge \ldots \wedge var_k = value_k$ と矛盾しない．
  (e) アクション a による変数の更新 $var_i := {value_i}^{after}$ が $var_1 = {value_1}^{after} \wedge \ldots \wedge \ldots \wedge var_i = {value_i}^{after} \wedge \ldots \wedge var_k = {value_k}^{after}$ に反映されている．

(2) e/a のとき
$(u_1 = q_1 \wedge \ldots \wedge u_n = q_n \wedge var_1 = value_1 \wedge \ldots \wedge var_k = value_k \wedge t_1 < c_1 < t_1\prime \wedge \ldots \wedge t_n < c_n < t_n\prime \wedge t_T < T < t_T\prime) \longrightarrow (u_1 = {q_1}^{after} \wedge \ldots \wedge u_n = {q_n}^{after} \wedge var_1 = {value_1}^{after} \wedge \ldots \wedge var_k = {value_k}^{after} \wedge {t_1}^{after} < c_1 < {t_1\prime}^{after} \wedge$

$\ldots \wedge {t_n}^{after} < c_n < {t_n}'^{after} \wedge {t_T}^{after} < T < {t_T}'^{after}$)．なお，$after$ は遷移後の状態名や変数値などを意味する記号である．ただし，以下を満たす：

(a) ${q_i}' = q_1 \times \ldots \times q_n$ および ${q_{i+1}}' = {q_1}^{after} \times \ldots \times {q_n}^{after}$．

(b) イベント e による，ある変数がある値であるといった条件 $var_i = value_i$ は $var_1 = value_1 \wedge \ldots \wedge var_i = value_i \wedge \ldots \wedge var_k = value_k$ と矛盾しない．

(c) アクション a による変数の更新 $var_i := {value_i}^{after}$ が ${var_1 = value_1}^{after} \wedge \ldots \wedge \ldots \wedge {var_i = value_i}^{after} \wedge \ldots \wedge {var_k = value_k}^{after}$ に反映されている．

■

Example 3（クロック遷移システムへの変換例）
図 3 のフラットな時間ステートチャートからクロック遷移システム $\mathbf{CTS} = (V, \Theta, \mathcal{T})$ を構成する．エアコン制御システムの時間ステートチャートは，以下のようにクロック遷移システムに変換できる：

(1) $V = D \cup C$ であり，D と C は以下のように定義できる．$D = \{u_1, u_2, message, is\_room\_hot, is\_door\_close, ac, VS\_assert\}$．$C = \{c_1, c_2, T\}$．ここで，$u_1$ と u_2 は $TEMP$ と $DOOR$ の状態を表す変数である．また，c_1 と c_2 はクロック変数であり，$TEMP$ のタイミング制約は c_1 で表現して，$DOOR$ のタイミング制約は c_2 で表現する．

(2) Θ は以下のように定義できる．
$\Theta = (u_1 = cool \wedge u_2 = open \wedge message = \perp \wedge is\_room\_hot = \perp \wedge is\_door\_close = \perp \wedge ac = \perp \wedge VS\_assert = \perp \wedge c_1 = 0 \wedge c_2 = 0 \wedge T = 0)$．
ただし，$\perp$ は未定義を意味する．

(3) 以下のように状態遷移関係 ρ_τ が定義される：

(a) $cool \times open$ から $hot \times open$ に状態遷移するとき
$(u_1 = cool \wedge u_2 = open \wedge message = \perp \wedge is\_room\_hot = \perp \wedge is\_door\_close = \perp \wedge ac = \perp \wedge VS\_assert = \perp \wedge 0 \leq c_1 \leq 5 \wedge 0 \leq c_2 \leq 5 \wedge 0 \leq T \leq 5) \wedge ({u_1}' = hot \wedge {u_2}' = open \wedge {message}' = room\_is\_hot \wedge {is\_room\_hot}' = 1 \wedge {is\_door\_close}' = \perp \wedge {ac}' = \perp \wedge {VS\_assert}' = \perp \wedge {c_1}' = 0 \wedge {c_2}' = 0 \wedge 0 \leq {T}' \leq 5)$．

(b) $hot \times open$ から $vhot \times open$ に状態遷移するとき
$(u_1 = hot \wedge u_2 = open \wedge message = room\_is\_hot \wedge is\_room\_hot = 1 \wedge is\_door\_close = \perp \wedge ac = \perp \wedge VS\_assert = \perp \wedge 0 \leq c_1 \leq 7 \wedge 0 \leq c_2 \leq 7 \wedge 0 \leq T \leq 7) \wedge ({u_1}' = vhot \wedge {u_2}' = open \wedge {message}' = room\_is\_hot \wedge {is\_room\_hot}' = 2 \wedge {is\_door\_close}' = \perp \wedge {ac}' = \perp \wedge {VS\_assert}' = \perp \wedge {c_1}' = 0 \wedge {c_2}' = 0 \wedge {T}' \geq 0)$．
.....
.....

■

4. 時間ステートチャートの安全性と活性の検証

時間ステートチャートが安全性や活性を充足することを検証するために，時間ステートチャートから構成したクロック遷移システムで演繹的証明する．この場合，時相論理の演繹的検証ルール[6]を使用する．

4.1 時相論理

まず，時相論理[12]の部分集合であり，公理系で証明する時相論理[6]の構文を定義する．

Definition 6（時相論理の構文）
時相論理式の集合は，以下の規則で生成される論理式の集合のうち最小のものである：

(1) 各原子命題 P は論理式である．ただし，原子命題はクロック遷移システムのシステム変数の集合への値の割当てにより定義される表明式である．

(2) p が論理式ならば，$\Box\, p$ は論理式である．

(3) p と q，r が論理式ならば，$\Box\,(p \rightarrow (qWr))$ は論理式である．

(4) p と r が論理式ならば，$\Box\,(p \rightarrow \Diamond r)$ は論理式である．

■

時相論理式の直感的意味としては以下のとおりである：$\Box\, p$ はつねに p が成り立つことを意味する．$\Box\,(p \rightarrow (qWr))$ はつねに p ならば r が成り立つまで q が成り立つことを意味する．$\Box\,(p \rightarrow \Diamond r)$ はつねに p ならばいつかは r が成り立つことを意味する．

次に，時相論理の意味を形式的に定義する．

Definition 7（時相論理の意味）
状態 s と論理式 p に対して，p が s 上で成り立つこ

とを $s \models p$ と書く．状態の無限列 $\sigma{=}s_0, s_1, s_2, \ldots$ をモデルとする．モデル σ に対して，位置 $j \geq 0$ で p が成り立つことを $\sigma, j \models p$ と書く．モデル上における論理式の充足関係は，以下のように定義できる：

(1) $\sigma, j \models p$ iff $s_j \models p$.

(2) $\sigma \models \Box\, p$ iff すべての j に対して $s_j \models p$ である．

(3) $\sigma \models \Box\, (p \rightarrow (qWr))$ iff すべての i に対して $s_i \models p$ ならば，すべての $j \geq i$ に対して $s_j \models r$，またはある $k \geq i$ に対して $s_k \models r$ かつすべての j ($i \leq j < k$) に対して $s_j \models q$ である．

(4) $\sigma \models \Box\, (p \rightarrow \Diamond r)$ iff すべての i に対して $s_i \models p$ ならば，$j \geq i$ に対して $s_j \models r$ である．

■

4.2 演繹的検証

安全性や活性の検証ルールは Kesten らによって開発されている[6]．文献 6) では，安全性として不変性と waiting-for 性質の検証ルールが提案されており，本論文では紙面の都合上から，不変性の検証例のみを以下に示す．また，他の安全性や活性の検証についても不変性と同様に，時間ステートチャートをクロック遷移システムに変換して，クロック遷移システムから各検証ルールの表明式を作成し，検証ルールの充足性を判定すれば検証できる．なお，他の文献 13) に，それらの具体例の一部を示す．

まず，不変性の検証ルールを定義して，次にその検証例を示す．

Definition 8（不変性の検証ルール）
表明 φ と p に対して，

1. $\Theta \rightarrow \varphi$
2. $\varphi \rightarrow p$
3. $\rho_\tau \wedge \varphi \rightarrow \varphi\prime\ (\forall \tau \in \mathcal{T})$
4. $- - - - - - - - - - - - - -$
5. $\Box p$

これは不変性を証明するルールであり，ルールの各行は，以下を意味する．1 行目は初期条件 Θ ならば表明 φ であることを意味する．2 行目は表明 φ ならば表明 p であることを意味する．3 行目は，$\forall \tau \in \mathcal{T}$ に対して，φ を保存することを意味する．4 行目は前提と結論を分離するラインである．5 行目は結論 $\Box p$ であり，$\Box$ はつねに成り立つことを意味する時相オペレータである．■

前例のエアコン制御システムにより，演繹的証明による不変性の検証の例を示す．

Example 4（演繹的証明による不変性の検証例）
前例のエアコン制御システムにより，演繹的証明による不変性の検証の例を以下に示す．ここで，$\Theta = (u_1 = cool \wedge u_2 = open \wedge message = \bot \wedge is\_room\_hot = \bot \wedge is\_door\_close = \bot \wedge ac = \bot \wedge VS\_assert = \bot \wedge t_1 = 0 \wedge t_2 = 0 \wedge T = 0)$ であり，$\varphi = p = \neg(u_1 = hot \wedge u_2 = open \wedge message = open\_door \wedge is\_room\_hot \geq 1 \wedge is\_door\_close = 0 \wedge ac = 1 \wedge VS\_assert = on \wedge t_1 \geq 0 \wedge t_2 \geq 0 \wedge T \geq 0)$ とする．これは，部屋の温度が高くてもドアが開いていると，つねにエアコンが動作しないことを意味する．

まず，1 行目は，$(u_1 = cool \wedge u_2 = open \wedge message = \bot \wedge is\_room\_hot = \bot \wedge is\_door\_close = \bot \wedge ac = \bot \wedge VS\_assert = \bot \wedge t_1 = 0 \wedge t_2 = 0 \wedge T = 0) \rightarrow \neg(u_1 = hot \wedge u_2 = open \wedge message = open\_door \wedge is\_room\_hot \geq 1 \wedge is\_door\_close = 0 \wedge ac = 1 \wedge VS\_assert = on \wedge t_1 \geq 0 \wedge t_2 \geq 0 \wedge T \geq 0)$ が成り立つので，$\Theta \rightarrow \varphi$ も成り立つ．

次に，2 行目は，$\varphi = p$ なので，$\varphi \rightarrow p$ は成り立つ．

次に，3 行目は，$\forall \tau \in \mathcal{T}$ に対して，$\rho_\tau \wedge \varphi \rightarrow \varphi\prime$ であり，以下に示す．

(1) $(u_1 = cool \wedge u_2 = open \wedge message = \bot \wedge is\_room\_hot = \bot \wedge is\_door\_close = \bot \wedge ac = \bot \wedge VS\_assert = \bot \wedge 0 \leq t_1 \leq 5 \wedge 0 \leq t_2 \leq 5 \wedge 0 \leq T \leq 5) \wedge (u_1\prime = hot \wedge u_2\prime = open \wedge message\prime = room\_is\_hot \wedge is\_room\_hot\prime = 1 \wedge is\_door\_close\prime = \bot \wedge ac\prime = \bot \wedge VS\_assert\prime = \bot \wedge t_1\prime = 0 \wedge t_2\prime = 0 \wedge 0 \leq T\prime) \wedge \neg(u_1 = hot \wedge u_2 = open \wedge message = open\_door \wedge is\_room\_hot \geq 1 \wedge is\_door\_close = 0 \wedge ac = 1 \wedge VS\_assert = on \wedge t_1 \geq 0 \wedge t_2 \geq 0 \wedge 0 \leq T) \rightarrow \neg(u_1\prime = hot \wedge u_2\prime = open \wedge message\prime = open\_door \wedge is\_room\_hot\prime \geq 1 \wedge is\_door\_close\prime = 0 \wedge ac\prime = 1 \wedge VS\_assert\prime = on \wedge t_1\prime \geq 0 \wedge t_2\prime \geq 0 \wedge T\prime \geq 0)$.

これは，$(u_1 = cool \wedge u_2 = open \wedge \ldots) \wedge (u_1\prime = hot \wedge u_2\prime = open \wedge message\prime = room\_is\_hot \wedge \ldots) \wedge \neg(u_1 = hot \wedge u_2 = open \wedge \ldots) \rightarrow \neg(u_1\prime = hot \wedge u_2\prime = open \wedge message\prime = open\_door \wedge \ldots)$ なので成り立つ．

(2) $(u_1 = hot \wedge u_2 = open \wedge message = room\_is\_hot \wedge is\_room\_hot = 1 \wedge is\_door\_close = \bot \wedge ac = \bot \wedge VS\_assert = \bot \wedge 0 \leq t_1 \leq 7 \wedge 0 \leq t_2 \leq 7 \wedge 0 \leq T \leq 7) \wedge (u_1\prime = vhot \wedge u_2\prime = open \wedge message\prime = room\_is\_hot \wedge is\_room\_hot\prime = 2 \wedge is\_door\_close\prime = \bot \wedge ac\prime = \bot \wedge VS\_assert\prime$

$=\perp \wedge t_1\prime = 0 \wedge t_2\prime = 0 \wedge T\prime \geq 0) \wedge \neg(u_1 = hot \wedge u_2 = open \wedge message = open\_door \wedge is\_room\_hot \geq 1 \wedge is\_door\_close = 0 \wedge ac = 1 \wedge VS\_assert = on \wedge t_1 \geq 0 \wedge t_2 \geq 0 \wedge T \geq 0) \rightarrow \neg(u_1\prime = hot \wedge u_2\prime = open \wedge message\prime = open\_door \wedge is\_room\_hot\prime \geq 1 \wedge is\_door\_close\prime = 0 \wedge ac\prime = 1 \wedge VS\_assert\prime = on \wedge t_1\prime \geq 0 \wedge t_2\prime \geq 0 \wedge T\prime \geq 0)$.

これは，$(u_1 = hot \wedge u_2 = open \wedge message = room\_is\_hot \wedge \ldots) \wedge (u_1\prime = vhot \wedge u_2\prime = open \wedge \ldots) \wedge \neg(u_1 = hot \wedge u_2 = open \wedge message = open\_door \wedge \ldots) \rightarrow \neg(u_1\prime = hot \wedge u_2\prime = open \wedge \ldots)$ なので成り立つ．

. . . . . .

. . . . . .

以下同様に，$\forall \tau \in \mathcal{T}$ に対して，$\rho_\tau \wedge \varphi \rightarrow \varphi\prime$ が成り立つことが示せる．

以上より，$\Box p$ が成り立つ． ■

5. 時間ステート チャートの詳細化検証

Kesten らのクロック遷移システムに関する演繹的証明系[6] では，安全性や活性の公理系は存在するが，詳細化の公理系は存在しない．本論文では，時間ステートチャートから構成したクロック遷移システムに関する詳細化検証の公理系を開発する．

5.1 詳細化検証の公理系

クロック遷移システムに関する詳細化検証の公理系では，具体化仕様の機能とタイミング制約が抽象化仕様の機能とタイミング制約に包含されれば，具体化仕様が抽象化仕様を詳細化していると考える．機能の包含性は通常の論理学的な含意であり，タイミング制約の包含性はクロック変数が表現する時間領域が包含されていると考える．たとえば，具体化仕様のクロック変数 c^A のタイミング制約が $l^A \leq c^A \leq u^A$ であり，抽象化仕様のそれが $l^C \leq c^C \leq u^C$ とすると，$l^A \leq l^C \leq u^C \leq u^A$ であれば，タイミング制約が包含されるとする．

Definition 9（詳細化の検証ルール）

$CTS^C = (V^C, \Theta^C, \mathcal{T}^C)$ が $CTS^A = (V^A, \Theta^A, \mathcal{T}^A)$ を詳細化していることを検証するルールを以下に示す．

1. $\Theta^C \rightarrow \Theta^A[\alpha]$
2. $\rho_{\tau^C} \rightarrow \bigvee_{\tau^A \in \mathcal{T}^A} \rho_{\tau^A}[\alpha]$ ($\forall \tau^C \in \mathcal{T}^C$ に対して)
3. −−−−−−−−−−−−−−−−−−−−−−−−
4. $CTS^C \sqsubseteq CTS^A$

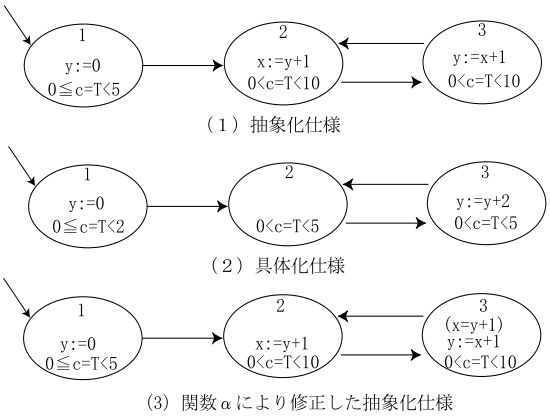

図 4 詳細化検証の例（1 対 1 の場合）
Fig. 4 Example of refinement verification (refinement mapping one to one).

これは詳細化を証明するルールであり，ルールの各行は，以下を意味する．1 行目は初期状態集合 Θ^C ならば $\Theta^A[\alpha]$ であることを意味する．2 行目は ρ_{τ^C} の状態遷移が抽象的な状態遷移 $\rho_{\tau^A}[\alpha]$ により説明されることを意味する．3 行目は前提と結論を分離するラインである．4 行目は結論 $CTS^C \sqsubseteq CTS^A$ であり，具体化仕様が抽象化仕様を正しく詳細化していることを意味する．ここで，α は V^A の変数を V^C の変数に置き換える写像または抽象化仕様の状態 Σ^A を具体化仕様の状態 Σ^C に置き換える写像を意味する． ■

まず，具体化仕様の状態と抽象化仕様の状態が 1 対 1 の場合の詳細化検証の事例を説明して，次に，具体化仕様の状態と抽象化仕様の状態が多対 1 の場合の詳細化検証の事例を説明する．

Example 5 （詳細化検証の例（1 対 1 の場合））

図 4 のように，具体化仕様 $CTS^C = (V^C, \Theta^C, \mathcal{T}^C)$ および抽象化仕様 $CTS^A = (V^A, \Theta^A, \mathcal{T}^A)$ が与えられたとする．ここで，α を以下のように定義する：

$$x = \begin{cases} \text{if 具体化仕様の状態が 3 then } y + 1 \\ \qquad\qquad \text{else } y \dot{-} 1 \end{cases}$$

ただし，$\dot{-}$ は通常の $-$ であるが，$y = 0$ のときは $y \dot{-} 1 = 0 \dot{-} 1 = 0$ とする．

以下では，具体化仕様が抽象化仕様を正しく詳細化していることを検証する．

まず，$\Theta^C \rightarrow \Theta^A[\alpha]$ が成り立つことを示す．ここで，$\Theta^C = (u_1 = 1 \wedge y = 0 \wedge c = 0 \wedge T = 0)$ および $\Theta^A[\alpha] = (u_1 = 1 \wedge y = 0 \wedge c = 0 \wedge T = 0)$ なので，$\Theta^C \rightarrow \Theta^A[\alpha]$ は成り立つ．

次に，$\forall \tau^C \in \mathcal{T}^C$ に対して，$\rho_{\tau^C} \rightarrow \bigvee_{\tau^A \in T^A} \rho_{\tau^A}[\alpha]$ が成り立つことを示す．

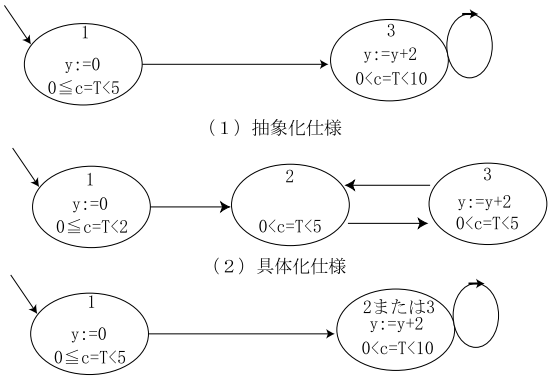

図 5 詳細化検証の例（多対 1 の場合）
Fig. 5 Example of refinement verification (rifinement mapping many to one).

(1) 具体化仕様が状態 1 から状態 2 に遷移するとき $(u_1 = 1 \wedge y = 0 \wedge 0 \leq c < 2 \wedge 0 \leq T < 2) \wedge (u_1\prime = 2 \wedge y\prime = y \wedge 0 \leq c\prime < 5 \wedge 0 \leq T\prime < 5) \rightarrow (u_1 = 1 \wedge y = 0 \wedge 0 \leq c < 5 \wedge 0 \leq T < 5) \wedge (u_1\prime = 2 \wedge y\prime = y \wedge 0 \leq c\prime < 10 \wedge 0 \leq T\prime < 10)$ である．具体化仕様の機能とタイミング制約が抽象化仕様の機能とタイミング制約に包含されるので，上記論理式は成り立つ．

(2) 具体化仕様が状態 2 から状態 3 に遷移するとき $(u_1 = 2 \wedge y = y\prime \wedge 0 < c < 5 \wedge 0 < T < 5) \wedge (u_1\prime = 3 \wedge 0 < c\prime < 5 \wedge 0 < T\prime < 5) \rightarrow (u_1 = 2 \wedge y = y\prime \wedge 0 < c < 10 \wedge 0 < T < 10) \wedge (u_1\prime = 3 \wedge x =$ if 具体化仕様の状態が 3 then $y + 1$ else $y \dot{-} 1 = y\prime + 1 \wedge 0 < c\prime < 10 \wedge 0 < T\prime < 10)$ である．具体化仕様の機能とタイミング制約が抽象化仕様の機能とタイミング制約に包含されるので，上記論理式は成り立つ．

(3) 具体化仕様が状態 3 から状態 2 に遷移するとき $(u_1 = 3 \wedge 0 < c < 5 \wedge 0 < T < 5) \wedge (u_1\prime = 2 \wedge y\prime = y + 2 \wedge 0 < c\prime < 5 \wedge 0 < T\prime < 5) \rightarrow (u_1 = 3 \wedge 0 < c < 10 \wedge 0 < T < 10) \wedge (u_1\prime = 2 \wedge y\prime = x + 1 \wedge x =$ if 具体化仕様の状態が 3 then $y + 1$ else $y \dot{-} 1 = y + 1 \wedge 0 < c\prime < 10 \wedge 0 < T\prime < 10)$ である．具体化仕様の機能とタイミング制約が抽象化仕様の機能とタイミング制約に包含されるので，上記論理式は成り立つ．

上記 (2) と (3) を繰り返すので，$\forall \tau^C \in \mathcal{T}^C$ に対して，$\rho_{\tau^C} \rightarrow \bigvee_{\tau^A \in \mathcal{T}^A} \rho_{\tau^A}[\alpha]$ が成り立つ．

以上より，具体化仕様が抽象化仕様を正しく詳細化していることが検証できた．

■

Example 6（詳細化検証の例（多対 1 の場合））

図 5 のように，具体化仕様 $CTS^C = (V^C, \Theta^C, \mathcal{T}^C)$ および抽象化仕様 $CTS^A = (V^A, \Theta^A, \mathcal{T}^A)$ が与えられたとする．ここで，α を以下のように定義する：

$$u_1 = \begin{cases} \text{if 具体化仕様の状態} = 2 \text{ then } 3 \\ \qquad \text{else 具体化仕様の状態} \end{cases}$$

ただし，u_1 は抽象化仕様の状態である．α は，具体化仕様の状態が 2 ならば抽象化仕様の状態が 3 であり，具体化仕様の状態が 2 でないならば抽象化仕様と具体化仕様の状態が等しいことを意味する．

以下では，具体化仕様が抽象化仕様を正しく詳細化していることを検証する．

まず，$\Theta^C \rightarrow \Theta^A[\alpha]$ が成り立つことを示す．ここで，$\Theta^C = (u_1 = 1 \wedge y = 0 \wedge c = 0 \wedge T = 0)$ および $\Theta^A = (u_1 = 1 \wedge y = 0 \wedge c = 0 \wedge T = 0)$ である．Θ^A の状態 $u_1 = 1$ は抽象化仕様の状態なので，α により具体化仕様の状態に変換する．ところで，α の定義により，具体化仕様の状態が 2 でなければ抽象化仕様と具体化仕様の状態名は等しい．ゆえに，$\Theta^A[\alpha] = (u_1 = if$ 具体化仕様の状態 $= 2\ then\ 3\ else$ 具体化仕様の状態 $= 1 \wedge y = 0 \wedge c = 0 \wedge T = 0)$ なので，$\Theta^C \rightarrow \Theta^A[\alpha]$ は成り立つ．

次に，$\forall \tau^C \in \mathcal{T}^C$ に対して，$\rho_{\tau^C} \rightarrow \bigvee_{\tau^A \in \mathcal{T}^A} \rho_{\tau^A}[\alpha]$ が成り立つことを示す．

(1) 具体化仕様が状態 1 から状態 2 に遷移するとき $(u_1 = 1 \wedge y = 0 \wedge 0 \leq c < 2 \wedge 0 \leq T < 2) \wedge (u_1\prime = 2 \wedge y = y\prime \wedge 0 \leq c\prime < 5 \wedge 0 \leq T\prime < 5) \rightarrow (u_1 =$ if 具体化仕様の状態 $= 2$ then 3 else 具体化仕様の状態 $= 1 \wedge y = 0 \wedge 0 \leq c < 5 \wedge 0 \leq T < 5) \wedge (u_1\prime =$ if 具体化仕様の状態 $= 2$ then 3 else 具体化仕様の状態 $= 3 \wedge y\prime = y \wedge 0 \leq c\prime < 10 \wedge 0 \leq T\prime < 10)$ である．具体化仕様の機能とタイミング制約が抽象化仕様の機能とタイミング制約に包含されるので，上記論理式は成り立つ．

(2) 具体化仕様が状態 2 から状態 3 に遷移するとき $(u_1 = 2 \wedge 0 < c < 5 \wedge 0 < T < 5) \wedge (u_1\prime = 3 \wedge y\prime = y \wedge 0 < c\prime < 5 \wedge 0 < T\prime < 5) \rightarrow (u_1 =$ if 具体化仕様の状態 $= 2$ then 3 else 具体化仕様の状態 $= 3 \wedge 0 < c < 10 \wedge 0 < T < 10) \wedge (u_1\prime =$ if 具体化仕様の状態 $= 2$ then 3 else 具体化仕様の状態 $= 3 \wedge y\prime = y \wedge 0 < c\prime < 10 \wedge 0 < T\prime < 10 \wedge u_1\prime = u_1)$ である．具体化仕様の機能とタ

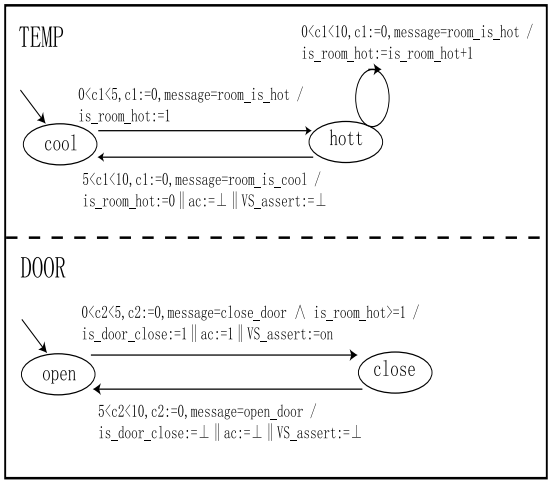

図 6 エアコン制御システムの抽象化仕様
Fig. 6 Abstract specification of aircon control system.

イミング制約が抽象化仕様の機能とタイミング制約に包含されるので，上記論理式は成り立つ．

(3) 具体化仕様が状態 3 から状態 2 に遷移するとき $(u_1 = 3 \wedge y = 0 \wedge 0 < c < 5 \wedge 0 < T < 5) \wedge (u_1\prime = 2 \wedge y\prime = y + 2 \wedge 0 < c\prime < 5 \wedge 0 < T\prime < 5) \rightarrow (u_1 =$ if 具体化仕様の状態 $= 2$ then 3 else 具体化仕様の状態 $= 3 \wedge y = 0 \wedge 0 < c < 10 \wedge 0 < T < 10) \wedge (u_1 =$ if 具体化仕様の状態 $= 2$ then 3 else 具体化仕様の状態 $= 3 \wedge y\prime = y + 2 \wedge 0 < c\prime < 10 \wedge 0 < T\prime < 10 \wedge u_1\prime = u_1)$ である．具体化仕様の機能とタイミング制約が抽象化仕様の機能とタイミング制約に包含されるので，上記論理式は成り立つ．

上記 (2) と (3) を繰り返すので，$\forall \tau^C \in \mathcal{T}^C$ に対して，$\rho_{\tau^C} \rightarrow \bigvee_{\tau^A \in \mathcal{T}^A} \rho_{\tau^A}[\alpha]$ が成り立つ．
ゆえに，具体化仕様が抽象化仕様を正しく詳細化していることが検証できた．

■

5.2 詳細化検証の事例

エアコン制御システムにより，具体化仕様 $CTS^C = (V^C, \Theta^C, \mathcal{T}^C)$ が抽象化仕様 $CTS^A = (V^A, \Theta^A, \mathcal{T}^A)$ を詳細化していることの検証例を示す．

では，以下に，エアコン制御システムの詳細化検証を示す．ここでは，図 2 を具体化仕様，**図 6** を抽象化仕様とする．図 2 と図 6 では，状態図 $DOOR$ は等しく，状態図 $TEMP$ のみが詳細化されたとする．

以下では，図 6 の状態図 $TEMP$ が図 2 の状態図 $TEMP$ によって正しく詳細化されていることのみを検証する．

ここで，α を以下のように定義する：

$$u_1 = \begin{cases} \text{if 具体化仕様の状態が } hot \text{ または } vhot \\ \qquad \text{then } hott \\ \text{else 具体化仕様の状態} \end{cases}$$

ただし，u_1 は抽象化仕様の状態である．

まず，$\Theta^C \rightarrow \Theta^A[\alpha]$ が成り立つことを示す．ここで，$\Theta^C = (u_1 = cool \wedge message = \perp \wedge is\_room\_hot = \perp \wedge is\_door\_close = \perp \wedge ac = \perp \wedge VS\_assert = \perp \wedge c_1 = 0 \wedge T = 0)$ および $\Theta^A[\alpha] = (u_1 =$ if 具体化仕様の状態が hot または $vhot$ then $hott$ else 具体化仕様の状態 $= cool \wedge message = \perp \wedge is\_room\_hot = \perp \wedge is\_door\_close = \perp \wedge ac = \perp \wedge VS\_assert = \perp \wedge c_1 = 0 \wedge T = 0)$ である．以上より，$\Theta^C \rightarrow \Theta^A[\alpha]$ は成り立つ．

次に，$\forall \tau^C \in \mathcal{T}^C$ に対して，$\rho_{\tau^C} \rightarrow \bigvee_{\tau^A \in \mathcal{T}^A} \rho_{\tau^A}[\alpha]$ が成り立つことを示す．

(1) 具体化仕様が状態 $cool$ から状態 hot へ遷移するとき
$(u_1 = cool \wedge message = \perp \wedge is\_room\_hot = \perp \wedge is\_door\_close = \perp \wedge ac = \perp \wedge VS\_assert = \perp \wedge 0 \leq c_1 \leq 5 \wedge 0 \leq T \leq 5) \wedge (u_1\prime = hot \wedge message\prime = room\_is\_hot \wedge is\_room\_hot\prime = 1 \wedge is\_door\_close\prime = \perp \wedge ac\prime = \perp \wedge VS\_assert\prime = \perp \wedge c_1\prime = 0 \wedge 0 \leq T\prime \leq 5) \rightarrow (u_1 =$ if 具体化仕様の状態が hot または $vhot$ then $hott$ else 具体化仕様の状態 $= cool \wedge message = \perp \wedge is\_room\_hot = \perp \wedge is\_door\_close = \perp \wedge ac = \perp \wedge VS\_assert = \perp \wedge 0 \leq c_1 \leq 5 \wedge 0 \leq T \leq 5) \wedge (u_1\prime =$ if 具体化仕様の状態が hot または $vhot$ then $hott$ else 具体化仕様の状態 $= hott \wedge message\prime = room\_is\_hot \wedge is\_room\_hot\prime = 1 \wedge is\_door\_close\prime = \perp \wedge ac\prime = \perp \wedge VS\_assert\prime = \perp \wedge c_1\prime = 0 \wedge 0 \leq T\prime \leq 5)$ である．具体化仕様の機能とタイミング制約が抽象化仕様のそれに包含されるので，上記論理式は成り立つ．

(2) 具体化仕様が状態 hot から状態 $vhot$ へ遷移するとき
$(u_1 = hot \wedge message = room\_is\_hot \wedge is\_room\_hot = 1 \wedge is\_door\_close = \perp \wedge ac = \perp \wedge VS\_assert = \perp \wedge 0 \leq c_1 = 0 \leq 7 \wedge 0 \leq T) \wedge (u_1\prime = vhot \wedge message\prime = room\_is\_hot \wedge is\_room\_hot\prime = 2 \wedge is\_door\_close\prime = \perp \wedge ac\prime = \perp \wedge VS\_assert\prime$

$=\perp \wedge c_1\prime = 0 \wedge 0 \leq T\prime) \rightarrow (u_1 =$ if 具体化仕様の状態が hot または $vhot$ then $hott$ else 具体化仕様の状態 $= hott \wedge message = room\_is\_hot \wedge is\_room\_hot = 1 \wedge is\_door\_close = \perp \wedge ac = \perp \wedge VS\_assert = \perp \wedge 0 \leq c_1 \leq 10 \wedge 0 \leq T \leq 10) \wedge (u_1\prime =$ if 具体化仕様の状態が hot または $vhot$ then $hott$ else 具体化仕様の状態 $= hott \wedge u_2\prime = open \wedge message\prime = room\_is\_hot \wedge is\_room\_hot\prime = 2 \wedge is\_door\_close\prime = \perp \wedge ac\prime = \perp \wedge VS\_assert\prime = \perp \wedge c_1\prime = 0 \wedge 0 \leq T)$ である．具体化仕様の機能とタイミング制約が抽象化仕様のそれに包含されるので，上記論理式は成り立つ．

(3) 具体化仕様が状態 $vhot$ から状態 $vhot$ へ遷移するとき

$(u_1 = vhot \wedge message = room\_is\_hot \wedge is\_room\_hot = 2 \wedge is\_door\_close = \perp \wedge ac = \perp \wedge VS\_assert = \perp \wedge 0 \leq c_1 = 0 \leq 10 \wedge 0 \leq T) \wedge (u_1\prime = vhot \wedge message\prime = room\_is\_hot \wedge is\_room\_hot\prime = 3 \wedge is\_door\_close\prime = \perp \wedge ac\prime = \perp \wedge VS\_assert\prime = \perp \wedge c_1\prime = 0 \wedge 0 \leq T\prime) \rightarrow (u_1 =$ if 具体化仕様の状態が hot または $vhot$ then $hott$ else 具体化仕様の状態 $= hott \wedge message = room\_is\_hot \wedge is\_room\_hot = 2 \wedge is\_door\_close = \perp \wedge ac = \perp \wedge VS\_assert = \perp \wedge 0 \leq c_1 \leq 10 \wedge 0 \leq T \leq 10) \wedge (u_1\prime =$ if 具体化仕様の状態が hot または $vhot$ then $hott$ else 具体化仕様の状態 $= hott \wedge u_2\prime = open \wedge message\prime = room\_is\_hot \wedge is\_room\_hot\prime = 3 \wedge is\_door\_close\prime = \perp \wedge ac\prime = \perp \wedge VS\_assert\prime = \perp \wedge c_1\prime = 0 \wedge 0 \leq T\prime)$ である．具体化仕様の機能とタイミング制約が抽象化仕様のそれに包含されるので，上記論理式は成り立つ．

.....

.....

同様な方法で，すべての状態遷移について，具体化仕様の機能とタイミング制約が抽象化仕様の機能とタイミング制約に包含されることが示せる．

以上より，$\forall \tau^C \in \mathcal{T}^C$ に対して，$\rho_{\tau^C} \rightarrow \bigvee_{\tau^A \in \mathcal{T}^A} \rho_{\tau^A}[\alpha]$ が成り立つ．

ゆえに，具体化仕様が抽象化仕様を正しく詳細化していることが検証できた．

6. むすび

本論文では，まず，時間ステートチャート[2]を一般化して，その操作的意味をクロック遷移システム上で定義した．これにより，ハードリアルタイムシステムの機能や並行性，タイミング制約が統合的に仕様記述できた．次に，安全性や活性を時相論理式で仕様記述して，クロック遷移システムが時相論理式を充足するかどうかを，演繹的証明で検証する手法を示した．次に，詳細化の検証の公理系を開発して，時間ステートチャートの詳細化を検証した．本論文のように，機能モデルと動作モデルを統合的に仕様記述すると，データ領域や実行の無限性により，仕様は無限な状態空間を有する時間付きの状態遷移システムになる．ゆえに，モデル検査[14]は不可能であり，検証作業は演繹的証明に頼らざるをえない．演繹的証明は数学的訓練が必要な作業であり，大規模システムへの適用は困難である．しかし，プロセスの相互排他制御の検証などのように検証対象を限定すれば，小規模システムとして仕様記述できて，演繹的検証が実現できて実用性は高いと考えられる．以上より，今後の研究課題としては，以下が重要である．

(1) 表明の自動生成や証明の再利用，GUI の向上などにより，演繹的証明作業の効率的な支援を実現する．

(2) 抽象実行などの抽象化技術を適用することにより，時間ステートチャートの自動検証を実現する．

(3) 時間ステートチャートのモジュールを定義して，その意味を形式的に定義することにより，モジュール検証を実現する．

謝辞 本研究は文部省科研費基盤 C「時相論理と並行計算，オートマトンの統合化による自律性のある分散システムの設計支援（代表：山根智）」の援助のもとで実施されました．

参考文献

1) 情報処理学会：特集：リアルタイムシステム，情報処理学会誌，Vol.35, No.1, pp.11–54 (1994).
2) 山根 智：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，情報処理学会論文誌，Vol.35, No.12, pp.2640–2650 (1994).
3) Alur, R. and Dill, D.: Automata for Modeling Real-Time Systems, *Lecture Notes in Computer Science 443*, pp.322–335 (1990).
4) Alur, R., Courcoubetis, C. and Dill, D.: Model checking for real-time systems, *5th LICS*,

pp.414–425 (1990).
5) Lynch, N.A. and Attiya, H.: Using mapping to prove timing properties, *Distributed Computing*, No.6, pp.121–139 (1992).
6) Kesten, Y., Manna, Z. and Pnueli, A.: Verifying clocked transition systems, *Lecture Notes in Computer Science 1066*, pp.13–40 (1996).
7) Harel, D., Lachover, H., Naamad, A., Pnueli, A., et al.: STATEMATE: A Working Environment for the Development of Complex Reactive Systems, *IEEE Trans. SE*, Vol.16, No.4, pp.403–414 (1990).
8) Peron, A. and Maggiolo, A.: Transitions as Interrupts: A New Semantics for Timed Statecharts, *Lecture Notes in Computer Science 789*, pp.806–821 (1994).
9) Harel, H. and Politi, M.: *Modeling Reactive Systems with Statecharts*, p.258, McGraw-Hill (1998).
10) Garey, M.R. and Johnson, D.S.: *Computers and Intractability*, p.340, W.H. Freeman and Company (1979).
11) Alur, R., Courcoubetis, C., Dill, D., Halbwachs, N. and Wong-Toi, H.: An implementation of three algorithms for timing verification based on automata emptiness, *Proc. IEEE RTSS*, pp.157–166 (1992).
12) Manna, Z. and Pnueli, A.: *Temporal Verification of Reactive Systems*, p.512, Springer-Verlag (1995).
13) 山ノ口，山根：実時間システムの演繹的検証，電子情報通信学会研究報告，SS2000-6, pp.1–8 (2000).
14) Henzinger, T.A., Nicollin, X., Sifakis, J. and Yovine, S.: Symbolic model checking for real-time systems, *Proc. 7th LICS*, pp.394–406 (1992).

(平成 12 年 5 月 26 日受付)
(平成 13 年 3 月 9 日採録)

山根　智（正会員）

昭和 59 年京都大学大学院工学研究科修士課程修了．現在，鹿児島大学工学部助教授．リアルタイムシステムの演繹的検証と自動演繹的検証の研究に従事．EATCS や ACM，IEEE 等各会員．