設計自動化 72－18
(1994. 10. 28)

タイムペトリネットにおけるCTL記号モデル検査法について

大川 保吉
東京工業大学
情報工学科
okawa@cs.titech.ac.jp

米田 友洋
東京工業大学
情報工学科
yoneda@cs.titech.ac.jp

東京工業大学 工学部 情報工学科 〒152 東京都目黒区大岡山 2-12-1

あ ら ま し モデル検査とは状態グラフ等でモデル化されたシステムの振る舞いが、ある時相論理式を満たすかどうかを調べる検証手法である。この際、状態グラフの状態爆発という問題が生じる。その解決法の一つとして、状態の集合、あるいは状態集合間の遷移関係を論理関数を用いて表し、論理関数演算により与えられた式を満たす状態集合を求める記号モデル検査法がある。しかし、リアルタイムシステムの記号モデル検査法については、従来あまり研究されていない。そこで、本研究では、リアルタイムシステムのモデル化にタイムペトリネットを用い、仕様の記述にCTL式を用いた場合の記号モデル検査法似ついて考察する。リアルタイムシステムには状態数が無限個あるという問題があるため、与えられた式の評価値が同じである状態同士を一つの状態クラスにまとめ状態クラス数を有限化する工夫がある。本稿では整数時刻で代表させる手法、region graphを用いる手法について考察した。

和文キーワード タイムペトリネット,CTL, 記号モデル検査法,region graph

CTL symbolic model checking method for Time Petri Net

Yasukichi Okawa
Department of Computer Science
Tokyo Institute of Technology
okawa@cs.titech.ac.jp

Tomohiro Yoneda
Department of Computer Science
Tokyo Institute of Technology
yoneda@cs.titech.ac.jp

Department of Computer Science, Tokyo Institute of Technology 2-12-1 Ookayama Meguro-ku
Tokyo 152, Japan

Abstract Model checking is a technique that checks weather the behavior of the system modeled by state graph satisfies the given temporal logic formula.In this process,a problem called state explosion sometimes occurs.One solution to this problem is symbolic model checking where set of states that satisfy some temporal properties or transition relation between state sets are expressed by logical functions and model checking is performed by using operations between those logical functions.Although symbolic model checking is actively studied for verification of non-real time systems,few research projects to apply symbolic model checking to real time system verification are reported. In this paper,as a special case of real time system verification,we study symbolic model checking in the case that systems are described by time Petri nets and properties to be verified are expressed by CTL formulas.In order to obtain finite structures of time Petri nets,sets of states that are equivalent in the evaluation of formulas should be collapsed. We tried two different approaches,one is based on discrete firing time model and the other is based on region graph model.

英文 key words Time Petri nets,CTL,symbolic model checking,region graph

1 はじめに

モデル検査 [1] とは、状態グラフ等でモデル化されたシステムの振る舞いが、ある時相論理式を満たすかどうかを調べる検証手法である。この際、問題となるのはシステムの並列動作の増加に伴い、状態グラフの状態数が指数関数的に増加することである。この問題に対する一つの解決法として、状態グラフそのものを直接扱わず、ある性質を持った状態の集合、あるいは状態集合間の遷移関係を論理関数を用いて表し、論理関数演算を用いて与えられた式を満たす状態集合を求める記号モデル検査法 [2],[3] が提案されている。

一方、従来の記号モデル検査に関する研究はほとんどが非リアルタイムシステムについてであり、リアルタイムシステムの記号モデル検査に関する研究はまだそれほどされていない。本研究では [4] と同様な手法であるが、特にリアルタイムシステムのモデル化にタイムペトリネットを用い、仕様記述のために分岐時間論理の一種である CTL(Computation Tree Logic) を用いた場合の記号モデル検査法について具体的に述べる。

リアルタイムシステムでは、システムの状態の表現に実数値の時刻を必要とするため状態数が無限になる。そこで、リアルタイムシステムの検証では、全ての式の評価が等しい状態同士を一つの状態クラスにまとめて、状態クラス数を有限化する工夫が必要である。その方法としては、整数時刻で代表させる手法 [5]、region graph を用いる手法 [6]、連立不等式を用いる手法 [7],[8] 等があるが、本稿では前二者について考察する。なお、最後の不等式による手法はアルゴリズムに変更を加えることにより CTL モデル検査に適用可能であることが分かっており別稿において述べる予定である。

2 タイムペトリネット

[定義] 1 タイムペトリネット Nは
$N=(P,T,F,Eft,Lft,s_0)$ の 6つ組により定義される。

- P:プレースの有限集合
- T:トランジションの有限集合
- $F \subseteq (P\times T)\cup(T\times P)$:入出力関係
- $Eft : T \to \mathbf{N}$ 最小発火時間を与える関数
  ただし $\mathbf{N}$ は非負の整数集合
- $Lft : T \to \mathbf{N}$ 最大発火時間を与える関数
- s_0:初期 *marking* □

本研究では、簡単化のため各プレースに高々1 個のトークンしか存在しない **1-safe** タイムペトリネットのみを考える。トークンのあるプレースの集合を **marking** といい、トランジションの入力プレース全てにトークンがある時そのトランジションは **enabled** であるという。なおトランジション t の入力プレースの集合を$\bullet t$ と表し、出力プレースの集合を $t\bullet$と表す。

タイムペトリネットのトランジションには clock という非負実数をとる変数が付加されていると考える。この clock は、そのトランジションが enabled になったとき reset され (値が 0 にされ)、その後そのトランジションが enabled であり続ける限りは clock の値は reset されてから経過した時間に等しい値を示す。トランジション t_iの clock を c_iとする時、enabled なトランジション t_iは $Eft(t_i)\le c_i\le Lft(t_i)$ ならいつでも発火することが出来、$c_i > Lft(t_i)$ となることはない。すなわちトランジション t_iが継続的に enabled であれば、enabled となってから必ず $Eft(t_i)$ から $Lft(t_i)$ の間に発火する。marking s_{i-1}でトランジション t_iが発火すると $s_i=(s_{i-1}-\bullet t_i)\cup t_i\bullet$なる marking が得られる。

タイムペトリネットの run を次のように定義する。

[定義] 2 *run* rは次のような形をした有限または無限の系列である。

$r:\langle s_0,\tau_0\rangle \xrightarrow{t_1}\langle s_1,\tau_1\rangle \xrightarrow{t_2}\langle s_2,t_2\rangle\to\cdots$

- $t_i\in T$: 発火したトランジション
- $s_i\in 2^P$: t_iの発火により得られる *marking*(s_0は初期 *marking*)
- $\tau_i\in\mathbf{R}^+$: t_iの発火時刻 ($\tau_0=0$) □

$T=\{t_1,t_2,\cdots,t_{|T|}\}$ に対し $C=\{c_1,c_2,\cdots,c_{|T|}\}$ とする時、$C\to\mathbf{R}^+$を (clock への) 時間割当と呼び、$\Gamma(N)$ を時間割当の集合とする。タイムペトリネットの状態、および次状態を次のように定義する。

[定義] 3 タイムペトリネットの状態は $\langle s,\nu\rangle$ である。ただし、s は *marking*、$\nu\in\Gamma(N)$ である。タイムペトリネットは初期 *marking* s_0で全ての *enabled* な *clock* を *reset* した状態 $\langle s_0,\nu_0\rangle$ からスタートする。状態 $\langle s,\nu\rangle$ の次状態は以下の $\langle s',\nu'\rangle$ であり、$\langle s,\nu\rangle\to\langle s',\nu'\rangle$ と記す。

$$\langle s',\nu'\rangle=\begin{cases}\langle (s-\bullet t_i)\cup t_i\bullet,[E\mapsto 0]\nu\rangle \\ \qquad if\ t_i\text{は}s\text{で}enabled, Eft(t_i)\le\nu(c_i)\le Lft(t_i \\ \langle s,\nu+\tau\rangle \quad if\ \forall t_i(t_i\text{は}s\text{で}enabled)[(\nu+\tau)(c_i)\le Lft(t_i\end{cases}$$

ただし Eは t_iの発火により新たに *enabled* となったトランジションの集合、$[E\mapsto 0]\nu$は、Eに含まれるトランジションの *clock* のみを *reset* した以外はνと同じ時間割当、$\nu+\tau$ は $\forall c\in C[(\nu+\tau)(c)=\nu(c)+\tau]$ なる時間割当である。 □

3 CTL

3.1 syntax

CTL 式は次の帰納的定義により得ることのできる式集合である。

1. $true, false$ および原子命題は CTL 式である。
2. ϕ, ψが CTL 式であるならば、$\neg\phi$や$\phi \wedge \psi$や$\phi \vee \psi$も CTL 式である。
3. ϕ, ψが CTL 式であるならば、$\mathbf{AX}\phi, \mathbf{A}[\phi \mathcal{U} \psi]$ も CTL 式である
4. ϕ, ψが CTL 式であるならば、$\mathbf{EX}\phi, \mathbf{E}[\phi \mathcal{U} \psi]$ も CTL 式である

3.2 semantics

structure $M = (S, R, L)$ に関して semantics を定義する。S, R, L はそれぞれ

- S ： 状態の有限集合
- $R \subseteq S \times S$ ： 遷移関係
- L ： 状態に対して原子命題集合を割り当てる関数

である。

1. $s_i \models true, s_i \not\models false$
2. $s_i \models p \Leftrightarrow p \in L(s_i)$
3. $s_i \models \phi \wedge \psi \Leftrightarrow s_i \models \phi$かつ $s_i \models \psi$
4. $s_i \models \phi \vee \psi \Leftrightarrow s_i \models \phi$または $s_i \models \psi$
5. $s_i \models \mathbf{AX}\phi \Leftrightarrow \forall s'[s'|(s_i, s') \in R] \models \phi$
6. $s_i \models \mathbf{EX}\phi \Leftrightarrow \exists s'[s'|(s_i, s') \in R] \models \phi$
7. $s_i \models \mathbf{A}[\phi \mathcal{U} \psi] \Leftrightarrow \forall \pi(\pi = s_i s_{i+1} \cdots)[\exists n(n$ は $n \geq i$ なる自然数$)[s_n \models \psi$かつ $\forall j(i \leq j < n)[s_j \models \phi]]]$
8. $s_i \models \mathbf{E}[\phi \mathcal{U} \psi] \Leftrightarrow \exists \pi(\pi = s_i s_{i+1} \cdots)[\exists n(n$ は $n \geq i$ なる自然数$)[s_n \models \psi$かつ $\forall j(i \leq j < n)[s_j \models \phi]]]$

タイムペトリネット Nのプレースの集合 Pを原子命題の集合とし、プレース pにトークンがあるとき $p \in L(s)$ とする。4 節では

$$S = \{\langle s_i, \tau_i \rangle | \langle s_0, \tau_0 \rangle \xrightarrow{t_1} \cdots \xrightarrow{t_i} \langle s_i, \tau_i \rangle \xrightarrow{t_{i+1}} \cdots \text{は } N\text{run}\},$$
$$\exists t_i[\langle s_{i-1}, \tau_{i-1} \rangle \xrightarrow{t_i} \langle s_i, \tau_i \rangle] \iff (\langle s_{i-1}, \tau_{i-1} \rangle, \langle s_i, \tau_i \rangle) \in R$$

に基づく structure 上で CTL 式を解釈し、5 節では

$$S = \{\langle s, \nu \rangle | \langle s, \nu \rangle \text{ は } N\text{の状態 }\},$$
$$\langle s, \nu \rangle \to \langle s', \nu' \rangle \iff (\langle s, \nu \rangle, \langle s' \nu' \rangle) \in R$$

に基づく structure 上で CTL 式を解釈する。タイムペトリネット Nと CTL 式ϕのモデル検査とは、$\langle s_0, 0 \rangle \models \phi$あるいは、$\langle s_0, \nu_0 \rangle \models \phi$の真偽値を求めることである。

4 整数時刻発火モデル

タイムペトリネットは発火時刻として全ての非負実数値を取ることができるので、その run は無限にある。もし発火時刻を整数時刻のみに限ることができれば、その run の数は有限のものとなりリアルタイムシステムの検証は容易になる。ここで [5] によれば、ある時相論理的性質は全ての run に対して検査しても、発火時刻を整数時間に限った run($\tau_i \in \mathbf{N}$ という制約を与えたもの、以下 discrete-run と呼ぶ) のみに対して検査しても同じとなることが証明されている。そこでこの方法が我々の目指す CTL のモデル検査に適用できるか検討してみる。[5] より以下の補題が容易に証明できる。

[補題] 1 任意の *run* は、それと *marking* 系列が全く同じ *discrete-run* を持つ □

これにより、discrete-run 集合はトランジションの発火系列だけを見れば全ての run の集合と等価であることが分かる。従って、例えば線形時相論理式は discrete-run 集合で評価しても正しい結果が得られる。

一方、CTL のような分岐時間論理では、式の評価は run ではなく、発火によって生じる分岐のある木構造に対して行われる。よって discrete-run による木と通常の run による木が等価な構造を持つことを示す必要がある。

ここで図 1のタイムペトリネットを考える。トランジション t_1, t_4は $[Eft, Lft] = [0, 1]$ のトランジションであり、それ以外のものは全て発火時刻が決定的である。また、このネット中ではそれぞれ $(t_2, t_3), (t_5, t_6), (t_7, t_8)$ が競合しており、そのどちらかのみが発火できる。$0 < c_1 < 1$ かつ $0 < c_4 < 1$ かつ $c_1 + c_4 = 1$ で発火する run 集合を考えると、図の太線で示したような部分木構造になる。ここでそれぞれの競合について時間制約を考えたうえで発火できるものを考えてみると、$(t_2, t_3), (t_5, t_6)$ のうちそれぞれ t_3, t_6しか発火できない[1]が、(t_7, t_8) の競合ではどちらも発火できる。ところが、このような部分木構造は discrete-run 集合の木構造には存在しない。すなわち、t_2, t_5が発火できない discrete-run では、t_8も発火できない。従って例えば $\phi = \mathbf{E}[\{(p_1 \to \neg\mathbf{EF}p_2) \wedge (q_1 \to \neg\mathbf{EF}q_2)\}\mathcal{U}(r_1 \wedge r_2)]$ とすると、discrete-run 集合の木構造で評価は $\langle s_0, 0 \rangle \models \phi$は偽となるが、全ての run 集合の木構造では真となる。これによりこのような discrete-run によるモデルでは CTL モデル検査はできないことがわかる。

5 region graph によるモデル検査

[6] では時間オートマトンの TCTL モデル検査法が提案されている。TCTL は実時間に関する性質を記述できるように CTL を拡張した実時間論理である。この手法を用い

[1] $c_2 < Eft(t_2), c_5 < Eft(t_5)$ であるから

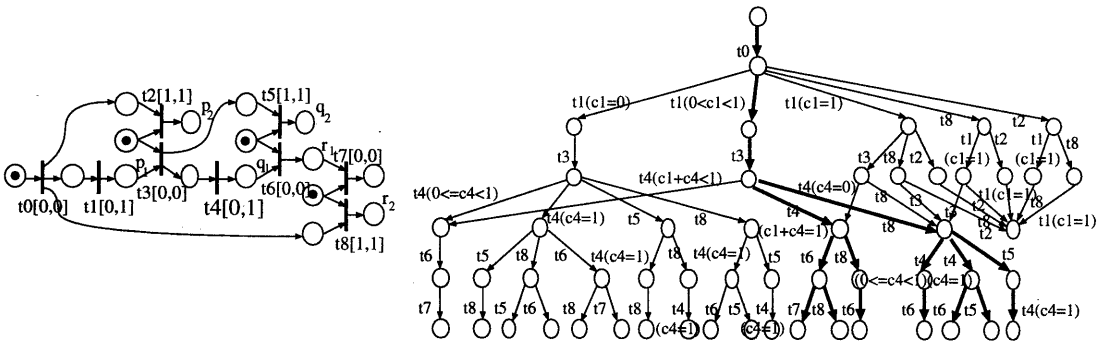

図 1: タイムペトリネットとその状態グラフ

れば、タイムペトリネットの CTL モデル検査も実現可能である。以下その概略を示す。

タイムペトリネットの状態数は、一般には無限となる。よってそれに対応する CTL structure の大きさも無限になる。しかしながこれら全ての状態が CTL 式の評価を異なるものにするわけではない。もし marking が同じで clock の時間割当はその整数部分が一致しており、小数部分はその順序において等しい状態があったとすれば、この 2 つを区別することの出来る CTL 式は存在しない。clock の整数部分はそのトランジションの発火条件を満たしているかどうかを判定するために必要であり、小数部分はどの clock の整数部分を次に更新すべきかを決定するのに必要である。

このような考え方に基づいて、等価な状態のクラスである region を以下のように定義する。

[定義] 4 時間割り当ての等価性

$\nu, \nu' \in \Gamma(N)$ が与えられたとき、次のような条件を満足するとき、かつ、そのときのみ $\nu \cong \nu'$ とする。

- $\forall x \in C, \lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$
- $\forall x, y \in C, fract(\nu(x)) \leq fract(\nu(y)) \iff fract(\nu'(x)) \leq fract(\nu'(y))$

ただし、$\lfloor a \rfloor$ は a の整数部分、$fract(a)$ は a の小数部分を表す。 □

[定義] 5 *region* は $\langle s, [\nu] \rangle$ である。ただし、s は *marking*、$[\nu]$ は ν が属する $\Gamma(N)$ の $\cong$ に関する *equivalence class* を表す。□

$\langle s, \nu \rangle$ によって表される状態が無限個あるのに対して、region の数は有限個である。従って region 同士のトランジションの関係を表す region graph は有限であり、region の定義によりこの region graph を CTL structure としてモデル検査を行っても状態グラフを用いた場合と同じ結果が得られる。

ただし、ここで region graph におけるトランジションにはタイムペトリネットのトランジションに加えて時間経過によるトランジションがある。トランジションは発火しておらず marking は変化していないが、時間割当が変化した場合である。このように時間が経過しただけのトランジションを τ-トランジションと呼ぶことにする。このような τ-トランジションが存在するため region graph 上の遷移系列はトランジションの発火しか考慮していない run に比べて遷移の数が多くなってしまう。そのため **AX**, **EX** オペレータのある CTL 式は 2 つの structure 上で異なる評価結果が得られる。そこで、ここでは **AX**, **EX** オペレータを認めないことにする。

6 記号モデル検査法

前節のように region graph を構成することによりタイムペトリネットにおける CTL 式のモデル検査を行うことができるが、region graph 中における region の数はネット中のトランジション数や clock の最大値に対して指数関数的に増加する。そこで、状態グラフを直接構成する必要のない記号モデル検査法を試みる。

6.1 特性関数

集合 $A \subseteq E$ に対し次の条件を満たす関数 $\mathcal{X}_A : E \to \{0,1\}$ を A の特性関数と呼ぶ。

$$\mathcal{X}_A(\boldsymbol{x}) = 1 \iff \boldsymbol{x} \in A$$

$B = \{0,1\}$ をブール変数の集合とする。このとき $E = B^n$ ならば、上記の $\mathcal{X}_A$ は論理関数と考えることができる。例として $E = B^2$ の空間を考えたとき $A = \{00, 11\}$ に対し A の特性関数は

$$\mathcal{X}_A(\boldsymbol{x}) = \overline{x_0} \cdot \overline{x_1} + x_0 \cdot x_1$$

となる。ここで $\boldsymbol{x} = x_0 x_1$ である。

6.2 region graph モデル上でのタイムペトリネットの状態遷移関数

まず、時間割当の equivalence class $[\nu]$ を 2 組の整数 (a, b) の列で表現する。

[定義] 6 $T = \{t_1, t_2, \cdots, t_{|T|}\}$ に対し
$C = \{c_1, c_2, \cdots, c_{|T|}\}$ とする。 *equivalence class* $[\nu]$ の整数表現は $\alpha = \{(a_1, b_1), \cdots, (a_{|T|}, b_{|T|})\}$ である。ここで、

- $a_i = \lfloor \nu(c_i) \rfloor$
- $b_i = fr(c_i)$

$fr(x) =$
$card(\{v | v < fract(\nu(x)), v \in \{fract(\nu(y)) | y \in C\} \cup \{0\}\})$
ただし、$card(\mathcal{X})$ は集合 $\mathcal{X}$ の要素数を返す関数で、$fr(x)$ は *clock* の小数部分の順番を返す。 □

以上のように定義したαは、$\cong$に関する equivalence class に求められる全ての性質を満たす。

これ以後 region は $\langle s, \alpha \rangle$ で表すものとする。

次に、プレース p_iに対する論理変数$\dot{p}_i$、および上記のαの a_i, b_iに対する論理変数ベクトル$\dot{\boldsymbol{a}}_i, \dot{\boldsymbol{b}}_i$を考える。ただし、着目している region において、プレース p_iにトークンが存在するときのみ、論理変数$\dot{p}_i$は 1 となるものとし、論理変数ベクトル$\dot{\boldsymbol{a}}_i, \dot{\boldsymbol{b}}_i$は a_i, b_iの 2 進数表記を取るものとする。

$$\boldsymbol{s} = (\dot{p}_1, \dot{p}_2, \cdots, \dot{p}_{|P|})$$
$$\boldsymbol{\alpha} = (\dot{\boldsymbol{a}}_1, \dot{\boldsymbol{b}}_1, \cdots, \dot{\boldsymbol{a}}_{|T|}, \dot{\boldsymbol{b}}_{|T|})$$

とする時、$\boldsymbol{s}\,\boldsymbol{\alpha}$を region $\langle s, \alpha \rangle$ の論理変数ベクトルとする。また、

$$\boldsymbol{s}' = (\dot{p}'_1, \dot{p}'_2, \cdots, \dot{p}'_{|P|})$$
$$\boldsymbol{\alpha}' = (\dot{\boldsymbol{a}}_1{}', \dot{\boldsymbol{b}}_1{}', \cdots, \dot{\boldsymbol{a}}_{|T|}{}', \dot{\boldsymbol{b}}_{|T|}{}')$$

に対して、$\boldsymbol{s}'\,\boldsymbol{\alpha}'$を$\boldsymbol{s}\,\boldsymbol{\alpha}$の次期 region に対する論理変数ベクトルとする。

region graph 上で、$\boldsymbol{s}\,\boldsymbol{\alpha}$に対応する region から$\boldsymbol{s}'\,\boldsymbol{\alpha}'$に対応する region に枝があるときのみ 1 となる論理関数 $Tr(\boldsymbol{s}\,\boldsymbol{\alpha}\,\boldsymbol{s}'\,\boldsymbol{\alpha}')$ を考える。このような遷移関数を region graph を構成することなく、タイムペトリネットから直接求めることができる必要がある。以下に、$Tr(\boldsymbol{s}\,\boldsymbol{\alpha}\,\boldsymbol{s}'\,\boldsymbol{\alpha}')$ をネットから直接求める方法について述べる。

$$\begin{aligned}
&Tr(\boldsymbol{s}\,\boldsymbol{\alpha}\,\boldsymbol{s}'\,\boldsymbol{\alpha}') = \\
&\bigwedge_{1 \le i \le |T|} \Big(\big(\bigwedge_{\dot{p} \in \bullet t_i} p \big) \rightarrow (\dot{\boldsymbol{a}}_i < Lft(t_i)) \Big) \wedge && (1) \\
&succ(\boldsymbol{\alpha}\,\boldsymbol{\alpha}') \wedge \bigwedge_{p \in P} (\dot{p} = \dot{p}') \ \vee && (2) \\
&\bigvee_{1 \le i \le |T|} \Big(\bigwedge_{p \in \bullet t_i} \dot{p} \wedge \bigwedge_{p \in t_i \bullet} \dot{p}' \wedge \bigwedge_{p \in (\bullet t_i - t_i \bullet)} \neg \dot{p}' \wedge \\
&\qquad \bigwedge_{p \notin (\bullet t_i \cup t_i \bullet)} (\dot{p} = \dot{p}') && (3) \\
&\wedge (Eft(t_i) \le \dot{\boldsymbol{a}}_i) \wedge (\dot{\boldsymbol{a}}_i < Lft(t_i) \vee \\
&\qquad (\dot{\boldsymbol{a}}_i = Lft(t_i) \wedge \dot{\boldsymbol{b}}_i = 0)) && (4) \\
&\wedge RESET(t_i, \boldsymbol{\alpha}\,\boldsymbol{\alpha}') \Big) && (5)
\end{aligned}$$

式 (1) の条件が満たされたなら式 (2) のτ-トランジションが可能になる。$succ(\boldsymbol{\alpha}\,\boldsymbol{\alpha}')$ は equivalence class αからτ-トランジションで equivalence class α'に遷移できるとき 1 を返すとする。式 (3) はトランジション t_iの発火による marking の変化を表し、式 (4) は t_iの発火のために満たすべき時間制約を表す。式 (5) はトランジション t_iを発火させ、新たに enabled となったトランジションの clock を reset した equivalence class α'を生成するためのもので、$RESET(t_i, \boldsymbol{\alpha}\,\boldsymbol{\alpha}')$ は、equivalence class αからトランジション t が発火し次期 region $\langle s', \alpha' \rangle$ になるとき 1 を返す論理関数である。

$succ(\boldsymbol{\alpha}\,\boldsymbol{\alpha}'), RESET(t_i, \boldsymbol{\alpha}\,\boldsymbol{\alpha}')$ は以下のように構成できる。

$$\begin{aligned}
&succ(\boldsymbol{\alpha}\,\boldsymbol{\alpha}') = \\
&\Big(\bigvee_{1 \le i \le |T|} (\dot{\boldsymbol{b}}_i = 0) \rightarrow \\
&\quad \bigwedge_{1 \le i \le |T|} ((\dot{\boldsymbol{a}}_i{}' = \dot{\boldsymbol{a}}_i) \wedge (\dot{\boldsymbol{b}}_i{}' = \dot{\boldsymbol{b}}_i + 1)) \Big) \wedge && (6) \\
&\Big(\neg \bigvee_{1 \le i \le |T|} (\dot{\boldsymbol{b}}_i = 0) \rightarrow && (7) \\
&\quad \bigwedge_{1 \le i \le |T|} \Big(\big(\bigwedge_{1 \le j \le |T|} (\dot{\boldsymbol{b}}_i \ge \dot{\boldsymbol{b}}_j) \big) \rightarrow (\dot{\boldsymbol{a}}_i{}' = \dot{\boldsymbol{a}}_i + 1) \wedge (\dot{\boldsymbol{b}}_i{}' = 0) \Big) \wedge \\
&\qquad \Big(\neg \bigwedge_{1 \le j \le |T|} (\dot{\boldsymbol{b}}_i \ge \dot{\boldsymbol{b}}_j) \rightarrow (\dot{\boldsymbol{a}}_i{}' = \dot{\boldsymbol{a}}_i) \wedge (\dot{\boldsymbol{b}}_i{}' = \dot{\boldsymbol{b}}_i) \Big) \Big) && (8)
\end{aligned}$$

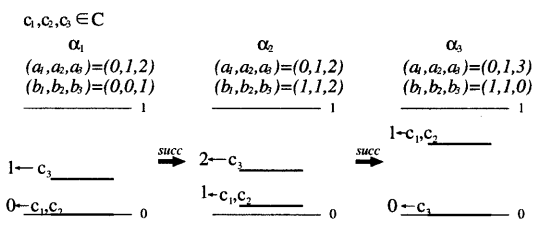

図 2: 関数 *succ* による equivalence calss の変化

式 (6) は clock の時間割当中に小数部分が 0 であるものがある場合、図 2の$\alpha_1 \rightarrow \alpha_2$のように、全ての clock の b_iを 1 増やし a_iはそのままにしておく。式 (8) は clock の時間割当中に小数部分が 0 であるものがない場合、図 2の$\alpha_2 \rightarrow \alpha_3$のように、$b_i$の値が最大である clock の整数部分を更新し、小数部分を 0 にする。

$$\begin{aligned}
&RESET(t_i, \boldsymbol{\alpha}\,\boldsymbol{\alpha}') = \\
&\quad (\boldsymbol{\alpha}^0 = \boldsymbol{\alpha}) \wedge \\
&\quad \bigwedge_{1 \le j \le |T|} \Big(((t_i \bullet \cap \bullet t_j \ne \emptyset) \rightarrow reset(t_j, \boldsymbol{\alpha}^{j-1}\,\boldsymbol{\alpha}^j)) \wedge \\
&\qquad ((t_i \bullet \cap \bullet t_j = \emptyset) \rightarrow (\boldsymbol{\alpha}^{j-1} = \boldsymbol{\alpha}^j)) \Big) \wedge \\
&\quad (\boldsymbol{\alpha}^{|T|} = \boldsymbol{\alpha}')
\end{aligned}$$

$RESET$はトランジション t_iが発火したことにより reset する必要がでた clock を、順次 reset していき正しい次期 equivalence class を求めるが、その過程で$\boldsymbol{\alpha}^1 \sim \boldsymbol{\alpha}^{|T|}$の論理変数を

一時的に用いる。$\boldsymbol{\alpha}^i$はc_1からc_iまでが正しく処理されたものである。よって式のように、$\boldsymbol{\alpha}' = \boldsymbol{\alpha}^{|T|}$となる。$reset(t_i, \boldsymbol{\alpha}\,\boldsymbol{\alpha}')$は equivalence class αの clock t_iを reset した equivalence class をα'とする論理式で、以下のようになる。

$$
\begin{aligned}
&reset(t_i, \boldsymbol{\alpha}\,\boldsymbol{\alpha}') = \\
&\quad (\dot{\boldsymbol{a}}_i{}' = 0) \wedge (\dot{\boldsymbol{b}}_i{}' = 0) \wedge \bigwedge_{t_j \in T - \{t_i\}} (\dot{\boldsymbol{a}}_j{}' = \dot{\boldsymbol{a}}_j) \wedge && (9)\\
&\quad \Big((\dot{\boldsymbol{b}}_i = 0) \rightarrow \bigwedge_{t_j \in T - \{t_i\}} (\dot{\boldsymbol{b}}_j{}' = \dot{\boldsymbol{b}}_j)\Big) \wedge && (10)\\
&\quad \Big((\dot{\boldsymbol{b}}_i \neq 0) \rightarrow \\
&\qquad \Big(\bigvee_{t_j \in T - \{t_i\}} (\dot{\boldsymbol{b}}_j = \dot{\boldsymbol{b}}_i) \rightarrow \bigwedge_{t_j \in T - \{t_i\}} (\dot{\boldsymbol{b}}_j{}' = \dot{\boldsymbol{b}}_j)\Big) \wedge && (11)\\
&\qquad \Big(\neg \bigvee_{t_j \in T - \{t_i\}} (\dot{\boldsymbol{b}}_j = \dot{\boldsymbol{b}}_i) \rightarrow && (12)\\
&\qquad\quad \bigwedge_{t_j \in T - \{t_i\}} \Big(((\dot{\boldsymbol{b}}_j > \dot{\boldsymbol{b}}_i) \rightarrow (\dot{\boldsymbol{b}}_j{}' = \dot{\boldsymbol{b}}_j - 1)) \wedge && (13)\\
&\qquad\qquad ((\dot{\boldsymbol{b}}_j < \dot{\boldsymbol{b}}_i) \rightarrow (\dot{\boldsymbol{b}}_j{}' = \dot{\boldsymbol{b}}_j))\Big)\Big)\Big) && (14)
\end{aligned}
$$

- (9) : t_iの clock を reset し、t_i以外の clock の整数部分は従来どおりとする
- (10) : t_iの小数部分が 0 であったならt_i以外の clock は従来どおり
- (11) : t_iと小数部分が等しい clock があったならt_i以外の clock は従来どおり
- (12) : それ以外なら、式 (13),(14) を行う。
- (13) : 小数部分がt_iより大きければ小数部分を 1 減らす
- (14) : 小数部分がt_iより小さければ従来どおり

6.3 論理関数による記号モデル検査法

$F_\phi(\boldsymbol{s}\,\boldsymbol{\alpha})$を CTL 式$\phi$を満たす region の集合を表す特性関数とする。$F_\phi(\boldsymbol{s}\,\boldsymbol{\alpha})$は、以下のように求めることができる。

1. $F_{true}(\boldsymbol{s}\,\boldsymbol{\alpha}) = 1$
2. $p_i \in P$とすると $F_{p_i}(\boldsymbol{s}\,\boldsymbol{\alpha}) = \dot{p}_i$
3. $F_{\neg\phi}(\boldsymbol{s}\,\boldsymbol{\alpha}) = \neg F_\phi(\boldsymbol{s}\,\boldsymbol{\alpha})$
4. $F_{\phi_1 \wedge \phi_2}(\boldsymbol{s}\,\boldsymbol{\alpha}) = F_{\phi_1}(\boldsymbol{s}\,\boldsymbol{\alpha}) \wedge F_{\phi_2}(\boldsymbol{s}\,\boldsymbol{\alpha})$
5. $F_0(\boldsymbol{s}\,\boldsymbol{\alpha}) = F_{\phi_2}(\boldsymbol{s}\,\boldsymbol{\alpha})$
   $F_{k+1}(\boldsymbol{s}\,\boldsymbol{\alpha}) =$
   $F_k(\boldsymbol{s}\,\boldsymbol{\alpha}) \vee (F_{\phi_1}(\boldsymbol{s}\,\boldsymbol{\alpha}) \wedge \neg\exists \boldsymbol{s}'\,\boldsymbol{\alpha}'(Tr(\boldsymbol{s}\,\boldsymbol{\alpha}\,\boldsymbol{s}'\,\boldsymbol{\alpha}') \wedge \neg F_k(\boldsymbol{s}'\,\boldsymbol{\alpha}')))$
   $F_{\mathbf{A}[\phi_1 \mathcal{U} \phi_2]} = F_\infty(\boldsymbol{s}\,\boldsymbol{\alpha})$
6. $F_0(\boldsymbol{s}\,\boldsymbol{\alpha}) = F_{\phi_2}(\boldsymbol{s}\,\boldsymbol{\alpha})$
   $F_{k+1}(\boldsymbol{s}\,\boldsymbol{\alpha}) =$
   $F_k(\boldsymbol{s}\,\boldsymbol{\alpha}) \vee (F_{\phi_1}(\boldsymbol{s}\,\boldsymbol{\alpha}) \wedge \exists \boldsymbol{s}'\,\boldsymbol{\alpha}'(Tr(\boldsymbol{s}\,\boldsymbol{\alpha}\,\boldsymbol{s}'\,\boldsymbol{\alpha}') \wedge F_k(\boldsymbol{s}'\,\boldsymbol{\alpha}')))$
   $F_{\mathbf{E}[\phi_1 \mathcal{U} \phi_2]} = F_\infty(\boldsymbol{s}\,\boldsymbol{\alpha})$

初期 region を $\langle s_0, \alpha_0 \rangle$ とすると

$$F_\phi(\boldsymbol{s_0}\,\boldsymbol{\alpha_0}) = 1 \iff \langle s_0, \alpha_0 \rangle \models \phi$$

であるから、F_ϕを求めることによりモデル検査が行える。

7 結論

タイムペトリネットの CTL モデル検査についていくつかの方法について考察した。まず整数時刻発火モデルについては、CTL 式のモデル検査には適さないことが分かった。region graph に基づく方法は CTL 式の **AX**, **EX** オペレータを除けばモデル検査に適用でき、region graph 上での遷移関数を論理関数により表現できることから、記号モデル検査法も適用可能であることが分かった。

今後は、以上の提案したアルゴリズムをインプリメントしたうえで、その有効性を実証するつもりである。

参考文献

[1] E. M. Clarke,E. A. Emerson,A. P. Sistla *Automatic verification of finite-state concurrent systems using temporal-logic specifications* ACM TOPLAS,Vol 8(2),244-263,1986

[2] K.L.McMillian,J.R.Burch,E.M.Clarke *Symbolic model checking:10^{20}states and beyond* Academic Press,98(2):142-170,1992

[3] 中根 清光 記号モデル検査法に基づく検証方式の効率化に関する研究 1994, 東京工業大学 修士論文

[4] Thomas A,Henzinger *Symbolic Model Checking for Real-time Systems* LICS'92,394-406,1992

[5] Tomas G.Rokicki *Representing and Modeling Circuits* PhD thesis,Stanford University,1993

[6] Rajee Alur,Costas Courcoubetis,David Dill *Model-Checking for Real-Time Systems* 5th IEEE LICS,414-425,1990

[7] B. Berthomieu and M. Diaz,*Modeling and verification of time dependent systems using time Petri nets* IEEE Trans. on Software Eng,Vol 17(3),259–273,1991

[8] T.Yoneda,A.Shibayama,H.Schlingloff,E.M.Clarke *Efficient Verification of Parallel Real-Time Systems* LNCS 697 Computer Aided Verification,321-332,1993