ソフトウェア基礎論　35－4
(1990. 5.24)

分岐時間正則時相論理とそのモデル検査アルゴリズム

濱口 清治　　平石 裕実　　矢島 脩三
京都大学工学部

順序機械など有限状態システムの設計が正しいことを保証するための検証手法の確立が重要な課題となっている。我々はこれまで 正則時相論理 (RTL, Regular Temporal Logic) のモデルチェックアルゴリズムを示し、またこれに基づき実際に順序機械の検証を行なってきた。

RTL は任意の有限オートマトンを特徴づけることができるが、 モデルチェックの計算複雑度は非初等的と大きい。 また、Clarke らによって提唱された CTL (Computational Tree Logic) は、有限状態システムの動作を反映する Kripke 構造 S の大きさ (Size(S)) と時相論理式 f の長さ (Len(f)) の積に対し線形時間でモデルチェックを行う ことができるが、表現能力が低く事象の繰り返しなどを必ずしも記述できない。

本稿では BRTL と その拡張である BRTL* を提案し、BRTL および BRTL* が CTL よりも真に高い表現能力を持つことを証明する。また、 BRTL に対するモデル検査アルゴリズムは CTL と同じく (Size(S)) と Len(f) の積に比例することを示す。

Branching Time Regular Temporal Logic and Its Model Checking Algorithm

Kiyoharu HAMAGUCHI, Hiromi HIRAISHI and Shuzo YAJIMA
Faculty of Engineering, Kyoto University

Verifying correctness of finite state systems like sequential machines is an important problem in system design. We have proposed Regular Temporal Logic (RTL) and have applied its model checking algorithm to formal verification of sequential machines.

Although RTL can characterize any behavior of sequential machines, the time complexity of the model checking algorithm is nonelementary. Computation Tree Logic (CTL) proposed by Clarke et.al has an model checking algorithm, which runs in time proportional to the product of the size of an given CTL formula and the size of a given Kripke structure. (Kripke structures reflect behaviors of finite state systems.) However, CTL is not powerful enough to express repetition of some events.

In this report, we propose branching time logics BRTL (Branching Time Regular Temporal Logic) and its extension BRTL*. It is proved that BRTL is more expressive than CTL and it has a model checking algorithm which runs in time proportional to the product of the size of a given Kripke structure and the length of a given formula.

1 はじめに

大規模集積回路技術の発達に伴い、設計の対象となる論理システムは、多様化、複雑化しつつある。設計された対象が設計者の課した仕様や条件を満足しているかどうかを確認するため、従来シミュレーションの手法が用いられてきたが、これは、必ずしも明確な数学的モデルに基づいてはおらず、したがってまた厳密に正しさが保証されないという問題がある。このため、時相論理、正則表現、CCS などのプロセス代数による対象の形式的記述および検証手法が研究されてきている。

時相論理 [1] は従来の述語論理や命題論理の体系に時間の様相を加えた論理体系であり、特に命題時相論理の種々の体系は充足可能性判定や、検証に応用が可能なモデル検査問題が決定可能という特徴を持っている。このため、検証過程で人手を介さない自動検証において有効と考えられ、順序機械、通信プロトコル、非同期回路、抽象化されたパイプライン設計などの形式的検証に応用され、成果をあげている [2, 3, 4, 5]。

命題時相論理としては、「常に」、「次の時点で」、「～が成り立つまで常に」という概念を表す演算子 $\Box$, $\bigcirc$, $\mathcal{U}$ を持つ古典的な時相論理 (Propositional Temporal Logic, 以下 PTL) があるが、Wolper が指摘したように [6]、有限オートマトンで記述可能な性質の内、たとえば、事象の繰り返しなどの性質は必ずしも記述することができない。

この難点を克服するため、Wolper は有限オートマトンおよび ω 有限オートマトンを時相演算子の記述に用いた ETL (Extended Temporal Logic, 以下 ETL) [6] を提案した。また我々は正則表現と等価な表現能力を持つ正則時相論理 RTL (Regular Temporal Logic, 以下 RTL) [7] を示し、順序機械の形式的検証に適用してきた [2]。

形式的検証手法には、設計された有限状態システムの動作を反映する Kripke 構造上で命題論理の論理式の真偽を判定するモデル検査の手法があるが、モデル検査問題の計算量を考えた場合、PTL, ETL では多項式領域完全 (PSPACE-complete)[8, 9]、RTL では非初等的 (nonelementary) [2] と極めて大きい。これらの論理体系は、原始命題への真偽の割当の系列に対して論理式の真偽が定まる線形時間モデルに基づいているが、一方、無限木の節点に対して原始命題への真偽の割当を定める分岐時間モデルに基づいた CTL (Computation Tree Logic) [3] では、Kripke 構造 S の大きさ Size(S) と時相論理式 f の長さ (Len(f)) の積に対し線形時間のモデル検査アルゴリズムが知られている。

しかしながら、CTL の演算子は $\Box, \bigcirc, \mathcal{U}$ のみであるため、有限状態システムの性質のうち、繰り返しなどの性質は十分に記述することができない。

本稿では、CTL より真に高い表現能力を持ち、かつ CTL と同じく Size(S) と Len(f) の積に比例する時間計算量を持つモデル検査アルゴリズムを持つ分岐時間正則時相論理 (Branching time Regular Temporal Logic, 以下 BRTL) を提案する。また、表現能力は BRTL と等価であるが記述量を少なくしうるよう拡張した BRTL* を示す。

BRTL は時相演算子として Büchi 型の ω 有限オートマトン [10] のサブクラスを用いており、BRTL* はオートマトン間にブール演算を許すよう拡張した時相論理である。

以下、2 章では、BRTL および BRTL* において、時相演算子として利用する Büchi 型の ω 有限オートマトンのサブクラスを定義し、その性質を明らかにした後、BRTL および BRTL* を定義する。3 章では、BRTL および BRTL* が CTL および CTL* よりも真に高い表現能力を持つことを証明する。4 章では、BRTL および BRTL* に対する モデル検査のアルゴリズムを示し、BRTL については、このアルゴリズムが Size(S) と Len(f) の積に比例する時間計算量を持つことを示す。

2 分岐時間正則時相論理 BRTL

本章では、分岐時間モデルに基づく時相論理 BRTL (Branching time Regular Temporal Logic) およびその拡張である BRTL* を定義する。

2.1 論理型決定性 ω 有限オートマトン

BRTL および BRTL* は時間概念を表現するための時相演算子の記述に、次に定義する ω 有限オートマトンを利用している。このオートマトンは、直観的には遷移の枝に、記号の代わりに命題論理式をラベル付けした決定性完全指定の ω 有限オートマトンであり、各原始命題への真偽値の割り当ての無限の列に対し、受理／非受理を決定するものになっている。

時相演算子を定義する際、他のクラスの有限オートマトンを用いることも考えられるが、これについては 4 章で計算量の議論を行なったのち、考察することとする。

以下では、命題論理における真と偽をそれぞれ T, F で表現する。命題論理式 f が 原始命題の集合 $P = \{p_1, \cdots, p_n\}$ から構成される時、各原始命題への真偽値の割り当て $v \in \{F, T\}^n$ に対する f の値を $f(v)$ と記す。

また、集合 X に対して、その要素の個数を $|X|$ と記述する。$A \Leftrightarrow B$ は A が B であるための必要十分条件であることを表している。また、X^ω および $X^\dagger$ によって X の要素の無限系列、有限系列の集合をそれぞれ表す。

定義 1 論理型決定性 ω 有限オートマトン *(*以下、*ldo-fa* と記す*)* $A = (Q, \Sigma, P, Br, \delta, q_0, F)$ を次のように定義する。

1. Q は有限個の状態の集合。
2. $\Sigma = \{F, T\}^n$ は入力ベクトルの集合。
3. $P = \{p_1, \cdots, p_n\}$ は原始命題の集合。BF を 原始命題の集合 P から構成される命題論理式の集合と定める。
4. $Br : Q \times Q \rightarrow BF$ は、状態の 2 つ組に命題論理式を割り当てる部分関数（したがって、命題論理式が割り当てられない状態の 2 つ組も存在しうる）

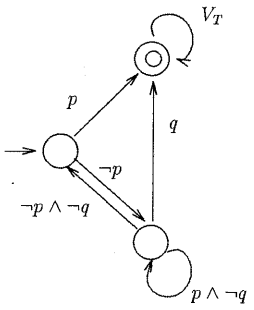

図 1: ldo-fa の例

である。ただし、Br は次の条件を満たす。任意の状態 q に対して $BF_q = \{f|$ ある q' が存在して、$Br(q,q') = f\}$ を考えた時、「任意の $f_1, f_2 \in BF_q$ に対して、$f_1 \wedge f_2 = F$」かつ「$\bigvee_{f \in BF_q} f$ は恒真関数」。

Edges(A) $\stackrel{\text{def}}{=}$ $\{(q,q')|$ ある f が存在して、$Br(q,q') = f\}$ とする。

$Br(Q \times Q) = \{f|$ ある q, q' が存在して、$Br(q,q') = f\}$ となる。

5. $\delta : Q \times \Sigma \to Q$ は状態遷移関数であり、次のように定義される。$q, q' \in Q$, $v \in \Sigma$ とする。
$\delta(q,v) = q' \Leftrightarrow f = Br(q,q')$ が 定義されていて、$f(v) = T$
この δ は、$\delta(q, v_0v_1 \cdots v_m) = \delta(\delta(v_0v_1 \cdots v_{m-1}), v_m)$ $(v_0, v_1, \cdots, v_m \in \Sigma)$ によって、系列に対しても拡張される。
6. q_0 は初期状態。
7. F は受理状態の集合である。$Q - F$ の要素を非受理状態と呼ぶ。

$Inf(\psi)$ を A に $\psi \in \Sigma^\omega$ を入力した時無限回通過する状態の集合とした時、A が受理する言語 を $\{\psi|Inf(\psi) \cap F$ が空でない $\}$ と定義し、$|A|$ と表現する。 □

図 1 に ldo-fa の例を示す。ラベルのない矢印が初期状態を指し示しており、◎は受理状態を表現している。p, q は原始命題である。

有限系列の受理／非受理を判定する決定性有限オートマトンの場合と異なり、受理状態と非受理状態を交換するという操作では、ldo-fa A に対して、$\Sigma^\omega - |A|$ を受理する ldo-fa を構成することは、一般にできない。

例を図 2 に示す。2 つの ldo-fa はともに $(TF)^\omega$ を受理する。

条件 1 *ldo-fa* $A = (Q, \Sigma, P, Br, \delta, q_0, F)$ を考える。A の任意 $q_r \in (Q - F)$ および任意の有限系列 $v_0v_1v_2 \cdots v_m$ $(v_i \in \Sigma, i = 0, 1, \cdots, v_m$ かつ $m \geq 1)$ に対して、ある $1 \leq b \leq m$ が存在して $v_0v_1 \cdots v_m = v_0v_1 \cdots v_bv_{b+1} \cdots v_m$

図 2: $(TF)^\omega$ を受理する ldo-fa

かつ $\delta(q_r, v_0v_1 \cdots v_b) = q_a$ かつ $\delta(q_r, v_{b+1} \cdots v_m) = q_r$ となるような $q_a \in F$ は存在しない。 □

上の条件 1 は、どの非受理状態 q_r からも、受理状態を経由して 再び q_r へ遷移させる入力系列が存在しないことを示している。

定義 2 条件 1 を満足する ldo-fa を ldo-fa-1 と呼ぶ。 □

このとき、次の補題が成立する。

補題 1 *ldo-fa-1* $A = (Q, \Sigma, P, Br, \delta, q_0, F)$ に対して、$\Sigma^\omega - |A|$ を受理する *ldo-fa-1* $\overline{A}$ は、A の受理状態と非受理状態を交換することによって得られる。

(証明) 条件 1 が成立すれば、どの受理状態 q_a からも、非受理状態を経由して 再び q_a へ遷移させる入力系列が存在しないことに注意する。

$\psi \in |A|$ ならば、$Inf(\psi) \subseteq F$ である。なぜなら、もしも、非受理状態 q_r が $Inf(\psi)$ に含まれるならば、$\psi = \psi_1\psi_2 \cdots \psi_i \cdots (\psi_i \in \Sigma^\dagger, i \geq 1)$ なる 無限個の ψ_i が存在して、$\delta(q_0, \psi_1) = q_r$ かつ $\delta(q_r, \psi_j) = q_r$ $(j \geq 2)$ となる。条件 1 より、q_r から q_r へ ψ_j によって遷移する際、受理状態を経由することはない。したがって、$Inf(\psi)$ は受理状態を含まないことになって、$\psi \in |A|$ に反する。ゆえに、$Inf(\psi) \cap (Q - F)$ は空。したがって、$\psi \notin |\overline{A}|$。

同様に $\psi \notin |A|$ ならば、$\psi \in |\overline{A}|$ も証明することができる。また、明らかに $\overline{A}$ は条件 1 を満足している。 □

ldo-fa-1 A の 受理状態と非受理状態を交換して得られる ldo-fa-1 を $\overline{A}$ と表す。

補題 2 二つの *ldo-fa-1* $A_i = (Q_i, \Sigma, P, Br_i, \delta_i, q_{i0}, F_i)$ $(i = 1, 2)$ に対して、$|A_1| \cup |A_2|$ を受理する *ldo-fa-1* $A_1|A_2 = (Q, \Sigma, P, Br, \delta, q_0, F)$ は次のように構成することができる。

- $Q = Q_1 \times Q_2 = \{(q_1, q_2)|q_1 \in Q_1 \ , \ q_2 \in Q_2\}$
- $Br : Q \times Q \to BF$ は次の条件を満たす。$q_i, q_i' \in Q_i$ $(i = 1, 2)$ とする。
$Br((q_1, q_2), (q_1', q_2'))$ が 定義されるのは、$Br_i(q_i, q_i')$ が、$i = 1, 2$ のどちらの場合にも定義されている時かつその時に限り、$Br((q_1, q_2), (q_1', q_2')) = Br_1(q_1, q_1') \wedge Br_1(q_2, q_2')$。
- $\delta : Q \times \Sigma \to Q$ は 定義 1にしたがい構成される。
- $q_0 = (q_{10}, q_{20})$。
- $F = \{(q_1, q_2)|q_1 \in F_1$ または $q_2 \in F_2\}$。

(証明) まず、Br が ldo-fa の条件を満足していることを示す。任意の状態 $q = (q_1, q_2) \in Q_1 \times Q_2$ に対して $BF_q = \{f |$ ある $q' = (q'_1, q'_2)$ が存在して、$f = f_1 \wedge f_2 = Br((q_1, q'_1), (q_2, q'_2))$, $f_1 = Br_1(q_1, q'_1)$, かつ $f_2 = Br_2(q_2, q'_2)\}$ を考えると、任意の $f = f_1 \wedge f_2, f' = f'_1 \wedge f'_2 \in BF_q$ に対して、Br_1 および Br_2 の定義より、$f \wedge f' = (f_1 \wedge f_2) \wedge (f'_1 \wedge f'_2) = F$

また、$q_i \in Q_i$ $(i = 1, 2)$ に対して、$BF_{q_i} = \{f_i |$ ある q'_i が存在して、$Br(q_i, q'_i) = f_i\}$ とすると、$\bigvee_{f \in BF_q} f = (\bigvee_{f_1 \in BF_{q_1}} f_1) \wedge (\bigvee_{f_2 \in BF_{q_2}} f_2)$ であり、これは恒真関数である。

また、ある非受理状態 $(q_1, q_2) \in (Q - F)$ に対して、ある受理状態 $(q'_1, q'_2) \in F$ があって、ある入力系列によって、(q_1, q_2) から (q'_1, q'_2) を遷移して (q_1, q_2) へ再び遷移したと仮定すると、q'_1 と q'_2 のいずれか一方は A_1 または A_2 の受理状態であることより、A_1 と A_2 が ldo-fa-1 であるという仮定に反する。したがって、$A_1|A_2$ は ldo-fa-1 である。 □

補題 1、2 より次の定理が導かれる。

定理 1 *ldo-fa-1* が受理する言語のクラスは、集合上のブール演算に関して閉じている。 □

2.2 BRTL および BRTL* の構文と意味

BRTL の論理式の真偽は Kripke 構造 (Kripke structure) に対して定まる。

$AP = \{p_1, p_2, \cdots, p_n\}$ を原始命題の集合とする。

定義 3 このとき、$S = \langle \Sigma, I, R, \Sigma_0 \rangle$ を Kripke 構造と呼ぶ。ここで、Σ は状態の集合、$I : \Sigma \to 2^{AP}$ は各状態に対し、原始命題の真偽を割り当てる関数、$R \subseteq \Sigma \times \Sigma$ は Σ 上の有向枝の集合 (任意の状態から少なくとも 1 本の枝が出ているとする) であり、Σ_0 は初期状態の集合である。

Kripke 構造の大きさ Size(S) を $|\Sigma|+|R|$ と定義する。 □

定義 4 構文

BRTL 式： 以下では、その集合を BF と記す。

$p \in AP$、$f, g \in BF$、また A がオートマトン演算子であれば、$p \in BF$、$(\neg f), (f \vee g) \in BF$、および $(\exists A) \in BF$ である。

オートマトン演算子

$B = \{f_1, f_2, \cdots, f_n\} \subseteq BF$ とする。このとき、$A[AP/B]$ は、A 中の各原始命題 $p_1, p_2, \cdots p_n \in AP$ を同時に それぞれ $f_1, f_2, \cdots, f_n$ で置き換えたものを表すとする。$A[AP/B]$ を A の B による代入例と呼ぶ。($A[AP/AP]$ は A に等しい。) この時、変化するのは $Br(q, q')$ の値である。

ある ldo-fa-1 A' および $B \subseteq BF$ が存在して、$A = A'[AP/B]$ ならば、A および $\neg A$ はオートマトン演算子である。

BRTL* 式： 以下では、その集合を BF* と記す。

BRTL 式においてオートマトン演算子の代わりに、次に定義する拡張オートマトン演算子を用いることを許したもの。

拡張オートマトン演算子

ある ldo-fa-1 A', A'_1, A'_2 および $B, B_1, B_2 \subseteq BF$ が存在して、$A = A'[AP/B]$, $A_1 = A'_1[AP/B_1]$, $A_2 = A'_2[AP/B_2]$ ならば、A、$(\neg A)$、および $(A_1 \vee A_2)$ は拡張オートマトン演算子である。 □

オートマトン演算子および拡張オートマトン演算子に対しても、ldo-fa-1 に対する演算 '-' および '|' を拡張する。$\overline{A}$ はオートマトン演算子 A の受理状態と非受理状態を交換したもの、$A_1|A_2 = A'_1[AP/B_1]|A'_2[AP/B_2]$ は補題 2 で述べた変換を、B_1 および B_2 の要素を原始命題とみなし機械的に適用した結果を表すものとする。

定義 5 意味

BRTL 式

Kripke 構造 $S = \langle \Sigma, I, R, \Sigma_0 \rangle$ に対して定義する。$S, s \models f$ は BRTL 式 f が状態 $s \in \Sigma$ に対して真であることを意味する。$p \in AP$、f と g は BRTL 式、A はオートマトン演算子とする。

- $S, s \models p$ iff $I(s) \ni p$
- $S, s \models (f \vee g)$ iff $S, s \models f$ または $S, s \models g$
- $S, s \models (\neg f)$ iff $S, s \not\models f$
- $S, s \models (\exists A)$ iff Kripke 構造 S 上の状態 s から始まる無限系列 $\sigma = s_0 s_1 s_2 \cdots$ と A の状態の無限系列 $q_0 q_1 q_2 \cdots$ があって次の条件を満足している。ここで $i = 0, 1, \cdots$ である。$f_i = Br(q_i, q_{i+1})$ かつ $S, s_i \models (f_i)$ かつ $q_0, q_1, \cdots$ の内、無限回出現する状態が全て A の受理集合に含まれる。
- $S, s \models \exists \neg A$ iff $S, s \models \exists \overline{A}$

BRTL* 式

BRTL 式の意味に次の項目を加えたもの。

- $S, s \models \exists(A_1 \vee A_2)$ iff $S, s \models \exists(A_1|A_2)$

全ての $s \in \Sigma_0$ に対し $S, s \models f$ の時、$S \models f$ と書く。 □

ブール演算子 $\wedge$, $\equiv$ および $\Rightarrow$ を通常の意味で用いるものとし、V_T は恒真式を表すとする。また $A_1 \wedge A_2 \stackrel{\text{def}}{=} \neg((\neg A_1) \vee (\neg A_2))$、$\forall A \stackrel{\text{def}}{=} \neg\exists\neg A$ および $\forall\neg A \stackrel{\text{def}}{=} \neg\exists A$ とする。$\forall(\exists)$ は、Kripke 構造上のすべての (ある) 経路を意味する経路限定子である。

単項演算子は二項演算子よりも高い優先順位を持つものとし、混乱の恐れのない時は '(' 、 ')' を省略することもある。

次に、BRTL* 式の長さを定義する。

定義 6 BRTL* 式 f および拡張オートマトン式 A に対して、部分式の集合 $\mathrm{SF}(f)$、$\mathrm{SF}(A)$、長さ $\mathrm{Len}(f)$ および $\mathrm{Len}(A)$ は次のように再帰的に定義される。

以下で、f, g, g_1, g_2 は BRTL* 式、A, A_1, A_2 は 拡張オートマトン式であるとする。

- f が原始命題の場合。$\mathrm{SF}(f) = \{f\}$。$\mathrm{Len}(f) = 1$。
- $f = \neg g$ の場合。それぞれ $\mathrm{SF}(f) = \mathrm{SF}(g) \cup \{\neg g\}$、$\mathrm{Len}(f) = \mathrm{Len}(g) + 1$。
- $f = g_1 \vee g_2$ の場合。それぞれ $\mathrm{SF}(f) = \mathrm{SF}(g_1) \cup \mathrm{SF}(g_2) \cup \{g_1 \vee g_2\}$、$\mathrm{Len}(f) = \mathrm{Len}(g_1) + \mathrm{Len}(g_2) + 1$。
- f が $\exists A$ の場合。$\mathrm{SF}(\exists A) = \mathrm{SF}(A) \cup \{\exists A\}$ および $\mathrm{Len}(f) = \mathrm{Len}(A)$。
- ある ldo-fa-1 $A' = (Q, \Sigma, P, Br, \delta, q_0, F)$ および $B \subseteq BF$ に対して $A = A'[AP/B]$ の場合。$\mathrm{SF}(A) = \bigcup_{g \in Br(Q \times Q)} \mathrm{SF}(g)$。
  $\mathrm{Len}(A) = |Q| + |\mathrm{Edges}(A')| + \sum_{f \in Br'(Q \times Q)} \mathrm{Len}(f) + \sum_{g \in B} \mathrm{Len}(g)$。ここで、$Br'(Q \times Q)$ は 代入前の A' にラベル付けされている命題論理式の集合を表している。
  すなわち、A' の状態数と枝の数、代入前の A' の枝にラベルされている命題論理式の長さの総和 と AP に 代入される BRTL* 式の長さの総和の合計。
- $A = \neg A_1$ の場合。$\mathrm{SF}(A) = \mathrm{SF}(A_1) \cup \{\neg A_1\}$ および $\mathrm{Len}(A) = \mathrm{Len}(A_1) + 1$。
- $A = A_1 \vee A_2$ の場合。$\mathrm{SF}(A) = \mathrm{SF}(A_1) \cup \mathrm{SF}(A_2) \cup \{A_1 \vee A_2\}$ および $\mathrm{Len}(A) = \mathrm{Len}(A_1) + \mathrm{Len}(A_1) + 1$。

□

3 BRTL の表現能力

本章では、BRTL の表現能力について述べる。

BRTL* における $A_1 \vee A_2$ の形式の拡張オートマトン演算子は、1.2 節の BRTL* の意味の定義から明らかなように、構文上の操作によって意味を変化させることなしに、BRTL 式 $A_1|A_2$ へ変換することができる。したがって、次の定理が成立する。

定理 2 任意の *BRTL** 式 f、任意の *Kripke* 構造 $S = \langle \Sigma, I, R, \Sigma_0 \rangle$ および $s \in \Sigma$ に対して、ある *BRTL* 式 f' が存在して、

$$S, s \models f \Leftrightarrow S, s \models f'$$

□

次に、CTL の定義の概略を示す。

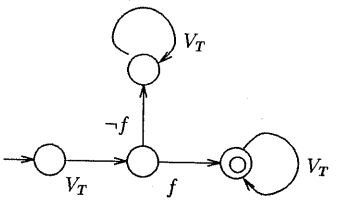

図 3: $Next(f)$

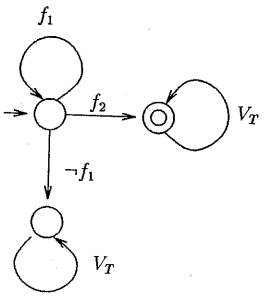

図 4: $Until(f_1, f_2)$

定義 7 CTL の論理式の真偽値は、Kripke 構造の各状態に対して定まる。

BRTL では経路限定子とオートマトン演算子によって、$\exists A$、$\exists \neg A$、$\exists(A_1 \vee A_2)$ の形式で時間の概念を表現しているが、CTL では、$\forall X f$、$\exists X f$、$\forall[f_1 \mathcal{U} f_2]$ および $\exists[f_1 \mathcal{U} f_2]$ のみが用いられる。これらの意味を次に示す。

- $S, s \models \forall X f$ ($\exists X f$) iff Kripke 構造 S 上の s から始まる全ての (ある) 無限系列 $s = s_0 s_1 s_2 \cdots$ に対して、$S, s_1 \models f$
- $S, s \models \forall[f_1 \mathcal{U} f_2]$ ($\exists[f_1 \mathcal{U} f_2]$) iff Kripke 構造 S 上の s から始まる全ての (ある) 無限系列 $s = s_0 s_1 s_2 \cdots$ に対して、ある $n \geq 0$ が存在して、$S, s_i \models f_1$ $(i = 0, 1, 2, \cdots, n)$ かつ $S, s_{n+1} \models f_2$

□

定理 3 任意の *CTL* 式 f、任意の *Kripke* 構造 $S = \langle \Sigma, I, R, \Sigma_0 \rangle$ および $s \in \Sigma$ に対して、ある *BRTL* 式 f' が存在して、

$$S, s \models f \Leftrightarrow S, s \models f'$$

(略証) 図 3 の $Next(f)$ および図 4 の $Until(f_1, f_2)$ をオートマトン演算子として用いると次に示すように CTL 式と等価な BRTL 式が構成できる。

- $\exists X f = \exists Next(f)$
- $\forall X f = \neg \exists \neg Next(f)$

- $\exists[f_1 \mathcal{U} f_2] = \exists Until(f_1, f_2)$
- $\forall[f_1 \mathcal{U} f_2] = \neg\exists\neg Until(f_1, f_2)$

□

CTL では経路限定子の有効範囲内で否定や和などの論理演算を使うことは許されていない。例えば $\exists((f_1 \mathcal{U} f_2) \vee Xf_3)$ や $\forall\neg(Xf)$ は CTL 式ではない。これらを記述できるように拡張した論理体系が CTL* である。

定理 4 任意の *CTL** 式 f、任意の *Kripke* 構造 $S = \langle \Sigma, I, R, \Sigma_0 \rangle$ および $s \in \Sigma$ に対して、ある *BRTL* 式 f' が存在して、

$$S, s \models f \Leftrightarrow S, s \models f'$$

(証明) BRTL* において、$Next(f)$ と $Until(f_1, f_2)$ を拡張オートマトン演算子として用いた場合を考えれば良い。 □

定理 4 は CTL* で表現できることは BRTL においてもまた表現できることを示している。次に、この逆は成立しないこと、すなわち、BRTL では表現可能であるが、CTL* では表現できない性質が存在することを示す。

以下の証明は、文献 [6] の定理 4.1 および系 4.2 を Kripke 構造に拡張したものになっている。

定義 8 Kripke 構造 $S = \langle \Sigma, I, R, \Sigma_0 \rangle$ と状態 $s_0 \in \Sigma$ に対して,

$$Re_i(S, s_0) = \{s_i | (s_j, s_{j+1}) \in R, j = 0, 1, 2, \cdots i-1\}$$

□

$Re_i(S, s_0)$ は s_0 から Kripke 構造 S 上の枝を i 回遷移して到達することのできる状態全ての集合を表している。

補題 3 p を原始命題とする。次の条件を満足する *Kripke* 構造 $S = \langle \Sigma, I, R, \Sigma_0 \rangle$ と状態の組 (S, s) の集合 を C_i と定義する。

1. $\forall s' \in Re_j(s).S, s' \models p \quad (j = 0, 1, \cdots, i)$,
2. $\forall s'' \in Re_{i+1}(s).S, s'' \models \neg p$,
3. $\forall s''' \in Re_j(s).S, s''' \models p \quad (j = i+1, i+2, \cdots)\}$

*CTL** 式 f が せいぜい n 個の 'X' しか含まないと仮定すると、$i, i' > n$ であるような任意の $C_i, C_{i'}$ から選んだ任意の $(S_i, s_i) \in C_i$, $(S_{i'}, s_{i'}) \in C_{i'}$ に対して、

$S_i, s_i \models f \Leftrightarrow S_{i'}, s_{i'} \models f$ (*)

すなわち、任意の $i > n$ に対して、f の真偽値は変化しない。

(証明) 部分式による帰納法によって証明する。

1. f が原始命題の場合は明らか。
2. f が $\neg g$ または $g_1 \vee g_2$ の場合は、g, g_1, g_2 に対して、帰納法の仮定より (*) が成立することから、g に対しても成立する。
3. f が $\forall Xg$ の場合。$(S, s) \in C_i$ に対して $S, s \models \forall Xg$ $\Leftrightarrow$ 全ての $(s, s') \in R$ であるような (S, s') に対して $S, s' \models \forall g$
   g に含まれる 'X' の個数はせいぜい $n-1$ 個、$(S, s') \in C_{i-1}$、$i-1 > n-1$ であることより、帰納法の仮定から、$S, s' \models \forall g$ か否かは、i に依存せずに決まる。ここで考えている (S, s') では、$S, s' \models \forall g$ か $S, s' \not\models \forall g$ かが、どの s' を選んでも同じであることより適当な $(s, s') \in R$ なる s' に対して $S, s \models \forall Xg \Leftrightarrow S, s' \models \forall g$
   故に、$S, s \models \forall Xg$ か否かは i に依存しない。
4. f が $\forall[g_1 \mathcal{U} g_2]$ の場合。$(S, s_i) \in C_i$ とする。
   $S, s_i \models \forall[g_1 \mathcal{U} g_2] \Leftrightarrow$
   s_i から始まる全ての無限系列 $s_i s_{i-1} s_{i-2} \cdots s_{n+1} \sigma_n$ ($s_j \in \Sigma$ σ_n は $(s_{n+1}, s_n) \in R$ なる状態 s_n から始まる S 上の状態の無限系列) に対して、
   $(S, s_i \models g_2)$ または $((S, s_i \models g_1)$ かつ
   $((S, s_{i-1} \models g_2)$ または $((S, s_{i-1} \models g_1)$ かつ
   $\cdots$
   $((S, s_{n+1} \models g_2)$ または $((S, s_{n+1} \models g_1)$ かつ
   $(S, s_n \models \forall[g_1 \mathcal{U} g_2]))) \cdots))$
   g_1, g_2 に対する帰納法の仮定より、
   $S, s_j \models g_k$ $(j = n+1, n+2, \cdots, i-1, i) \Leftrightarrow$
   $S, s_n \models g_k$ $(k = 1, 2)$
   故に、
   $S, s_i \models \forall[g_1 \mathcal{U} g_2]$ か否かは i に依存せずに定まる。
5. f が $\exists g$ の場合。$(S, s) \in C_i$ に対しては、$S, s \models \exists g$ $\Leftrightarrow S, s \models \forall g$ が成立することから、$\forall g$ の場合に帰着される。
6. f が $\forall\neg g$ の場合。
   $S, s \models \forall\neg g \Leftrightarrow S, s \models \neg\exists g \Leftrightarrow S, s \models \neg\forall g$
   故に、2. の場合に帰着する。
7. f が $\forall(g_1 \vee g_2)$ の場合。$S, s \models \forall(g_1 \vee g_2) \Leftrightarrow S, s \models \exists(g_1 \vee g_2) \Leftrightarrow S, s \models (\exists g_1) \vee (\exists g_2) \Leftrightarrow S, s \models (\forall g_1) \vee (\forall g_2)$
   故に、2. の場合に帰着する。

□

次の定理は、CTL* では 与えられた $m \geq 2$ に対して、Kripke 構造上の全ての (またはある) 経路 $s_0 s_1 \cdots$ において、s_i $(i = km, k = 0, 1, \cdots)$ に対し p が真になるという性質は記述できないことを示している。

定理 5 $m \geq 2$ が与えられたとする。*Kripke* 構造と状態の組 (S, s) の集合 $D = \{(S, s) | \forall s' \in Re_{km}.S, s' \models p, (k = 0, 1, \cdots)\}$ を考える。

$$(S, s) \in D \Leftrightarrow S, s \models f$$

を満足する *CTL** 式 f は存在しない。

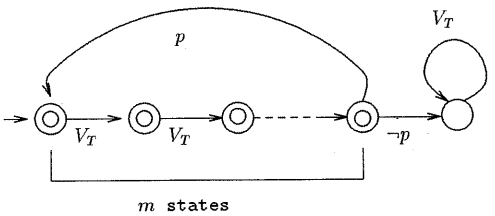

図 5: 繰り返し構造を記述するためのオートマトン演算子

(証明) 上記の条件を満足するような f が存在するとする。f が l 個の 'X' を含んでいるとする。

$km-1 \geq l$ とする。補題より任意の $(S,s) \in C_{km}$, $(S',s') \in C_{km-1}$ に対して、$S,s \models f \Leftrightarrow S',s' \models f$ となるが $(S,s) \in D$ かつ $(S',s') \notin D$ であることから、仮定に反する。 □

オートマトン演算子を用いれば、条件

$$S,s \in D \Leftrightarrow S,s \models f$$

を満足する BRTL 式 $f = \forall A_R$ は容易に構成することができる。A_R を図 5 に示す。

4 BRTL のモデル検査アルゴリズム

定義 9 Kripke 構造 $S = \langle \Sigma, I, R, \Sigma_0 \rangle$ BRTL 式 (BRTL* 式) f が与えられた時、$S \models f$ であるかどうかを判定する問題を、BRTL (BRTL*) のモデル検査問題 (model checking problem) と呼ぶ。 □

このモデル検査問題は、順序機械などの有限状態システムが BRTL 式で記述された条件を満足するかを検証するために用いることができる。例えば、(決定性の) 順序機械を対象とした場合には、順序機械の入出力が満たすべき性質を BRTL 式で記述し、順序機械をその動作を反映する Kripke 構造に変換した後、モデル検査を行なえば良い。順序機械を Kripke 構造に変換する手法については、文献 [2] を参照されたい。

モデル検査アルゴリズムを示す前に、正当性の証明に必要な定義および補題を示す。

以下では、特にことわりがない限り、Kripke 構造 $S = \langle \Sigma, I, R, \Sigma_0 \rangle$ と BRTL 式 $\exists A$ (ただし、ldo-fa-1 $A' = (Q, \Delta, P, Br, \delta, q_0, F)$ に対して、$A = A'[AP/B]$) を仮定する。

$q_0' \in Q$ にした $A'' = (Q, \Delta, P, Br, \delta, q_0', F)$ から得られる $A''[AP/B]$ を $A(q_0')$ と記すことにする。したがって、$A(q_0) = A$ になる。

BRTL の意味の定義と ldo-fa-1 の性質より次の補題が導かれる。

定義 10 Kripke 構造 S 上の状態と オートマトン演算子 A の状態の組の有限系列 $(s_0,q_0)(s_1,q_1)\cdots(s_n,q_n)$ を考える。

この時、条件 (**) をつぎのように定める。

条件 (**)

$(s_i, s_{i+1}) \in R$ $(i = 0, 1, \cdots n-1)$

かつ $S, s_i \models Br(q_i, q_{i+1})$ $(i = 0, 1, \cdots n-1)$ □

補題 4 $S,s \models \exists A \Leftrightarrow$

$(s_0,q_0)(s_1,q_1)\cdots(s_k,q_k)(s_{k+1},q_{k+1})\cdots(s_n,q_n)$

(ただし、$s_0 = s$) が存在して、条件 (**) および 次の条件 (***) を満足する。

条件 (***)

$s_k = s_n$ かつ $q_k = q_n$ かつ $q_j \in F$ $(j = k, k+1, \cdots n)$

□

つぎに BRTL のモデル検査アルゴリズムを示す。

アルゴリズム 1 モデル検査アルゴリズム

- 入力: Kripke 構造 S および BRTL 式 (BRTL* 式) f
- 出力: $S \models f$ ならば、yes。そうでなければ、no。
- 方法:

1. f 中の (拡張) オートマトン演算子を 2 章で示した方法により、'$\neg$' および '$\vee$' を含まない形に変換する。
2. 以下の (a)-(c) によって、f の部分式 $g_i = SF(f)$ に関して原始命題からボトムアップに、次の条件を満足する S の状態 s に g_i をラベル付けする。

$$S,s \models g_i$$

f まで終了した時点で、全ての $s_0 \in \Sigma_0$ に対して、f がラベルされていれば yes を、そうでなければ no を返す。

(a) g_i が 原始命題 の場合は明らか。

(b) g_i が $h_1 \vee h_2$ または $\neg h$ (h_1, h_2, h は BRTL 式) の場合、すでに S 上にラベル付けされている h_1, h_2, h を利用して、各状態での g_i の真偽を決定する。

(c) $g_i = \exists A$ の場合。

i. グラフ $G = (V, E)$ を構成する。ここで、
$V = \{(s,q) | s \in \Sigma, q \in Q\}$
$E = \{((s,q),(s',q')) | (s,s') \in R$ かつ $S, s \models Br(q,q')\}$

ii. G の節点集合を $V' = \{(s,q_F) | q_F \in F\}$ に制限した部分グラフ G' の強連結成分の節点集合を求め V_C とする。

iii. G 上で V_C のいずれかの節点に到達可能な節点の集合 V_R を求める。

iv. $(s,q_0) \in V_R$ (q_0 は A の初期状態) なる $s \in \Sigma$ に $\exists A$ をラベル付けする。

次にこのアルゴリズムの正当性を示す。

定理 6 *アルゴリズム 1 は $S \models f$ か否かを判定する。*

(証明) 部分式に関する帰納法によって証明する。

2.(a) の場合は明らか。2.(b) の場合、すでに h_1, h_2, h の S 上の各状態における真偽が確定していることより、g_i に対する真偽値の判定も正しく行うことができる。

2.(c) の場合、$(s, q_0) \in V_R$ が存在したとする。ii. と iii. より、系列

$(s, q_0)(s_1, q_1) \cdots (s_k, q_k)(s_{k+1}, q_{k+1}) \cdots (s_n, q_n)$

が存在して、$s_k = s$ かつ $s_n = q_n$ かつ $(s_k, q_k), \cdots, (s_n, q_n) \in V_C$ となる。

V_C の定義より、$q_k, q_{k+1}, \cdots q_n \in F$。したがって、補題 4 より、$S, s \models \exists A$。

$(s, q_0) \notin V_R$ かつ $S, s \models \exists A$ と仮定する。

補題 4 より、条件 $(**)$ を満足する系列

$(s, q_0)(s_1, q_1) \cdots (s_k, q_k)(s_{k+1}, q_{k+1}) \cdots (s_n, q_n)$

が存在する。少なくとも、$(s_k, q_k)(s_{k+1}, q_{k+1}) \cdots (s_n, q_n) \in V_C$ であるから、iii. より $(s, q_0) \in V_R$ となり仮定に反する。

$(s, q_0) \notin V_R$ ならば $S, s \not\models \exists A$ と仮定する。 □

定理 7 *BRTL に対するアルゴリズム 1 によるモデル検査の時間計算量は、* $O(\mathrm{Size}(S) \times \mathrm{Len}(f))$ *である。*

(証明)

1. では オートマトン演算子には、'¬' のみしか許されていないため、f 中に出現するオートマトン演算子の状態数の総和に比例する時間すなわち $O(\mathrm{Len}f)$ で、'¬' を含まない BRTL 式に変形することができる。

2.(b) では $\xi = h_1, h_2$ または h に対して $O(\mathrm{Size}(S) \times \mathrm{Len}(\xi))$ の時間で、S へのラベル付けが可能とすると、$h_1 \vee h_2$ または $\neg h$ に対してはどちらも $O(|\Sigma|)$ の時間でラベル付けができる。したがって、$g_i = h_1 \vee h_2$ または $\neg h$ に対し $O(\mathrm{Size}(S) \times \mathrm{Len}(g_i))$ の時間を要する。

2.(c) では、i. において、$S, s \models Br(q, q')$ を 判定するため、$Br(Q \times Q)$ の要素全て、すなわち A の枝にラベルされている BRTL 式すべてに対して、S へのラベル付けを行う。この時、まず、$\xi \in B$ に対するラベル付けを行い、次に、$Br(Q \times Q)$ の要素についてラベル付けを行うと、$O(\mathrm{Size}(S) \times (\Sigma_{f \in Br'(Q \times Q)} \mathrm{Len}(f) + \Sigma_{g \in B} \mathrm{Len}(g)))$ の時間で行なうことができる。

i. の G の $|V|$ は $O(|\Sigma| \times |Q|)$、$|E|$ は $O(|R| \times |\mathrm{Edges}(A)|)$ の大きさであることから、$Br(Q \times Q)$ の要素のラベル付けが終了していれば、G は $O(\mathrm{Size}(S) \times (|Q| + |\mathrm{Edges}(a)|)$ の時間で構成でき、$S, s \models Br(q, q')$ の判定と合わせて、$O(\mathrm{Size}(S) \times (\mathrm{Len}(A))$ の時間で構成できる。

ii. では文献 [11, Chap. 5.5] で示されている手法により、$O(\min(|V|, |E|))$ すなわち $O(\mathrm{Size}(S) \times (|Q| + |\mathrm{Edges}(a)|)$ の時間で V_C を求めることができる。

iii. および iv. もまた $O(\mathrm{Size}(S) \times (|Q| + |\mathrm{Edges}(a)|)$ の時間で V_R を求めることができる。

したがって、2.(c) は $O(\mathrm{Size}(S) \times (\mathrm{Len}(A))$ すなわち、$O(\mathrm{Size}(S) \times (\mathrm{Len}(g_i)))$ の時間で処理することができる。 □

系 1 *BRTL* を考える。与えられた BRTL 式 f に対して、f で用いられている各拡張オートマトン式 A_i $(0 \leq i \leq n)$ が A が $A_{i1}, A_{i2} \cdots A_{ik_i}$ から ¬ と ∨ を用いて構成されているとする。このとき、$k \stackrel{\mathrm{def}}{=} \max_{0 \leq i \leq n} k_i$ とすると、アルゴリズム 1 の時間計算量は $O(\mathrm{Size}(S) \times (\mathrm{Len}(f))^k)$ となる。*

(略証) アルゴリズムの 1. で $A_1 \vee A_2$ を変換する時に要する時間は、$A_1|A_2$ の構成から明らかなように、$O(\mathrm{Len}(A_1) \times \mathrm{Len}(A_2))$ となる。これと BRTL の場合の議論より、$O(\mathrm{Size}(S) \times (\mathrm{Len}(f))^k)$ の時間計算量が導びかれる。 □

BRTL で時相演算子を定義する際に用いた ldo-fa-1 は、Büchi 型 決定性 ω 有限オートマトンのサブクラスを用いているが、別のクラスの有限オートマトンを用いることについて、以下で考察する。

通常の有限オートマトン (有限系列を受理する) を用いれば、本稿と同様の形式で時相論理を構築することができる。しかし、有限系列を用いて条件を記述した場合、記述が繁雑になる場合があり [12]、ここでは、ω 有限オートマトンを採用している。

一般の Büchi 型 決定性 ω 有限オートマトンが受理する言語のクラスは集合上のブール演算に関して閉じていないことから [13]、論理体系に取り込むことは難しい。また、Büchi 型 非決定 ω 有限オートマトンのクラスは集合上のブール演算に関して閉じているが、その補集合を受理する ω 有限オートマトンはもとの状態数の指数オーダーの状態数を持つため、モデル検査問題のアルゴリズムとして計算量の小さいものを構築することが難しい。また、決定性 ω 有限オートマトンのクラスで Büchi 型 非決定 ω 有限オートマトンのクラスに等価なものとして、Muller 型 ω 有限オートマトン が知られているが、受理条件を状態集合の巾集合を用いて記述しなければならないことから、記述量が状態数の指数となる可能性があり実用上問題がある。

以上の事実を考慮して、オートマトン演算子として、補集合を受理するオートマトンへの変換が容易な ldo-fa-1 を用い、CTL より強力な表現能力を持つが、モデル検査アルゴリズムの時間計算量は CTL と同等の時相論理を構築している。

5 おわりに

本稿では、分岐時間モデルに基づく時相論理として、BRTL と その 拡張である BRTL* を提案した。またその表現能力が CTL より真に大きいことを証明し、さらに BRTL に関しては、式の長さと Kripke 構造の大きさの積に比例するモデル検査アルゴリズムを示した。

本稿で示した ldo-fa-1 は、満たさなければならない制約が必ずしも自明ではないことから、記述することはあまり簡単ではない。実用的には、構文上の制約により ldo-fa-1 のみが記述可能な言語を構築する必要がある。

順序回路を考えた場合、フリップフロップが n 個ならば、その状態数は 2^n のオーダーとなり、Kripke 構造のようなグラフ形式で表現すると状態数がフリップフロップの数の増加につれ、爆発的に増大する。これに対するアプローチとして、時相論理のモデル検査アルゴリズムを二分決定グラフを用いて行なう方法があり [5]、大規模な比較的規則性を持つ順序機械の検証に有効であることが明らかになっている。[5] では CTL を用いているが、BRTL にも適用が可能と考えられ、そのアルゴリズムの構築が今後の課題となっている。

謝辞: 種々ご議論頂いた高木直史博士、石浦菜岐佐氏を始め本学矢島研究室の皆様に深謝致します。

参考文献

[1] N. Rescher and A. Urquhart. *Temporal Logic.* Springer-Verlag, 1971.

[2] H. Hiraishi. Design Verification of Sequential Machines Based on a Model Checking Algorithm of ϵ-free Regular Temporal Logic. In *Computer Hardware Description Languages and their applications*, pages 249–263, June 1989.

[3] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications: A Practical Approach. In *10th ACM Symposium on Principles of Programming Languages*, pages 117–126, January 1983.

[4] D. L. Dill and E. M. Clarke. Automatic Verification of Asynchronous Circuits Using Temporal Logic. *IEE Proceedings*, 133:276–282, September 1986.

[5] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. In *Proceedings of 5th IEEE Symposium on Logic in Computer Science*, June 1990. to appear.

[6] P. Wolper. Temporal Logic Can Be More Expressive. In *Proceedings of 22nd Annual Symposium on Foundations of Computer Science*, pages 340–348, 1981.

[7] H. Hiraishi and S. Yajima. RTL:Regular Temporal Logic Expressively equivalent to regular set. *Journal of Information Processing Society of Japan*, pages 117–123, February 1987. In Japanese.

[8] A. P. Sistla and E. M. Clarke. The Complexity of Propositional Temporal Logic. *Journal of the ACM*, 32(3):733–749, July 1985.

[9] A. P. Sistla, M. Y. Vardi, and P. Wolper. The Complementation Problem for Büchi Automata with Applications to Temporal Logic. In *Proceedings of ICALP 85'*, pages 465–474, 1985.

[10] J. R. Büchi. On a Decision Method in Restricted Second Order Arithmetic. In *Logic Methodology and Philosophy of Science*, pages 1–12. Stanford University Press, 1962.

[11] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[12] K. Hamaguchi, H. Hiraishi, and S. Yajima. A Temporal Logic Expressively Equivalent to ω-Regular Set. Technical Report COMP88-8, IEICE, 1988. In Japanese.

[13] S. Eilenberg. *Automata, Languages, and Machines.* Academic Press, 1974.