リアクティブシステム実現可能性必要条件の判定手続き

冨田 尭

外部環境と継続的に相互作用するリアクティブシステムの高信頼化のために，その動作仕様を線形時相論理 LTL などで形式的に記述してその実現可能性を判定した上でシステムを自動合成する手法が研究されてきた．また，仕様が実現不能であった場合にその欠陥を効果的に分類するために，網羅性，戦略化可能性，保存可能性，安定化可能性の観点から階層的・体系的に定義される実現可能性必要条件群が導入されている．しかし，形式検証でしばしば用いられる ω-正規仕様に対する実現可能性必要条件判定手続きは，一部の必要条件にしか与えられておらず，近年新たに導入された実現可能性必要条件群の判定手続きが与えられていなかった．

本稿では，仕様欠陥分析を実施可能にするために，それらの実現可能性必要条件群に対する系統的な判定手続き群を与える．実現可能性必要条件群が体系的に整理されたことで，それらの判定手続きもいくつかの共通的操作の組合せとして構成できる．また，判定手続きの対称性に基いた新たな実現可能性必要条件及びその判定手続きも与える．そして，判定問題に対する計算量上界について考察する．

1 はじめに

1.1 背景

リアクティブシステム合成（reactive system synthesis）とは，利用者や外部システムなど環境（environment）と継続的に相互作用する**リアクティブシステム**を仕様から自動的に合成する手法である．リアクティブシステムは環境からの入力を制御できないため，与えられた**仕様**（specification）を満たすリアクティブシステムが存在するためには，その仕様は無矛盾，即ち，**充足可能**（satisfiable）なだけでは不十分であり，**実現可能**[26] [1] [29]（realizable）でなければならない．リアクティブシステム合成は，仕様の実現可能性判定も兼ねており，仕様が実現可能であるときにシステムを得ることができる．その際，仕様記述以外の工程では，人手を介さないため，不具合が混入する恐れがない．そのため，妥当な仕様さえ与えることができれば，仕様を実現するリアクティブシステムを確実に得ることができる．

しかしながら，実現可能性判定及び自動合成は一般に計算コストが非常に大きく，欠陥のある仕様を試行錯誤を通して実現可能かつ合理的になるよう修正・洗練することは容易ではない．そのため，森ら [25] 及び冨田ら [45] [34] は，充足可能性よりも強い実現可能性必要条件を階層的・体系的に提案した．

- 振舞性必要条件
  - **接頭的半強充足可能性** [45]（prefixically semi-strong satisfiability）
  - **接尾的半強充足可能性** [34]（suffixically semi-strong satisfiability）
  - **接尾的半真強充足可能性** [34]（suffixically semi-properly strong satisfiability）
  - **強充足可能性** [25]（strong satisfiability）
  - **真強充足可能性** [34]（properly strong satisfiability）
- 保存性必要条件
  - **保存的充足可能性** †1 [25]（preservably step-

Decision Procedures for necessary conditions of Reactive System Specifications.

Tomita, Takashi, 北陸先端科学技術大学院大学 情報社会基盤研究センター, Research Center for Advanced Computing Infrastructure, Japan Advanced Institute of Science and Technology.

†1 [25] [43] [38] [45] 用いられている「段階的」に代わりに，本稿では単に「保存的」という用語を用い，[34]

wise satisfiability）
  - **保存的強充足可能性** †1 [25]（preservably stepwise strong satisfiability）
  - **真保存的充足可能性** †1 [45]（properly preservably stepwise satisfiability）
- 安定性必要条件
  - **安定的充足可能性** †2 [34]（stably stepwise satisfiability）
  - **安定的強充足可能性** †2 [34]（stably stepwise strong satisfiability）

これらの必要条件充足判定を行うことで，仕様に欠陥がある，即ち，仕様が実現不能である際に，欠陥の分類や分析，欠陥箇所抽出 [18] により，効率的な仕様修正が可能となる．しかしながら，[34] で新たに導入された実現可能性必要条件群に対しては判定手続きが提案されていなかった．

1.2 目的

本稿では，仕様欠陥分析を実施可能にするために，これらの実現可能性必要条件群に対する系統的な判定手続き群を与える．

実現可能性必要条件群は階層的・体系的に定義されており，[25] [45] で導入された実現可能性必要条件の判定手続きはいくつかの共通的操作の組合せとして与えることができる．そこで，[25] [45] で導入された実現可能性必要条件の判定手続きをその共通的操作の組み合わせとして再整理するとともに，[34] で新たに導入された実現可能性必要条件にも判定手続きも同様の形で与える．

また，共通的操作の組み合わせとして再整理した結果として発見された手続き間の対称性に基づき新しい保存性必要条件

- **真保存的強充足可能性**（properly preservably stepwise strong satisfiability）

及びその判定手続きを与える．さらに判定問題の計算量上界を示す．

で用いられている「preservably stepwise」の訳として当てる．

†2 [34] で用いられている「stably stepwise」の訳として，本稿では単に「安定的」という用語を当てる．

2 準備

2.1 リアクティブシステム

外部環境と継続的に相互作用する開放系はしばしば**リアクティブシステム**としてモデル化される．受信した感知信号を基に制御信号を送信して対象を制御し続けるような制御器はその典型例である．非形式的には，リアクティブシステムは次のような特徴を持つ．

1. **入力非制御性**（Input uncontrollability）：リアクティブシステムは，自身から環境へ出力のみを制御でき，環境から自身への入力は制御できない．
2. **出力逐次性**（Output sequentiality）：各時刻の出力は，過去の振舞い（入出力列）のみから決定されなければならない．

ここで，リアクティブシステム・環境が制御する出力・入力イベントの出現を表す出力・入力命題をそれぞれ $\mathcal{O}$ と $\mathcal{I}$ と記す．形式的には，リアクティブシステムは次のように定義される．

定義 1 (リアクティブシステム) リアクティブシステムは，応答関数 $r : (2^{\mathcal{I}})^+ \to 2^{\mathcal{O}}$ である． ■

ここで，すべての可能なリアクティブシステムの集合を $\mathcal{R}$ と記す．また，環境からの無限入力列（つまり無限入力語）$\alpha = a_0 \cdot a_1 \cdots \in (2^{\mathcal{I}})^\omega$ に対する，リアクティブシステム r の無限出力列（つまり無限出力語）$r(a_0) \cdot r(a_0 \cdot a_1) \cdots \in (2^{\mathcal{O}})^\omega$ を $r^\omega(\alpha)$ と記す．そして，無限入力列 $\alpha = a_0 \cdot a_1 \cdots \in (2^{\mathcal{I}})^\omega$ と無限出力列 $\beta = b_0 \cdot b_1 \cdots \in (2^{\mathcal{O}})^\omega$ の各点和集合列（つまり無限入出力語）$(a_0 \cup b_0) \cdot (a_1 \cup b_1) \cdots \in (2^{\mathcal{I} \cup \mathcal{O}})^\omega$ を $[\alpha, \beta]$ と記す．

定義 2 (振舞い) リアクティブシステム $r \in \mathcal{R}$ の無限入力列 $\alpha \in (2^{\mathcal{I}})^\omega$ に対する振舞いは，各点和集合列 $[\alpha, r^\omega(\alpha)] \in (2^{\mathcal{I} \cup \mathcal{O}})^\omega$ である． ■

無限入力列 $\alpha \in (2^{\mathcal{I}})^\omega$ 及び無限出力列 $\beta \in (2^{\mathcal{O}})^\omega$ は，それぞれ $\alpha = \bar{\alpha} \cdot \hat{\alpha}$ 及び $\beta = \bar{\beta} \cdot \hat{\beta}$ であるような（1）任意有限長の**接頭語** $\bar{\alpha} \in (2^{\mathcal{I}})^*$ 及び $\bar{\beta} \in (2^{\mathcal{O}})^*$ と（2）無限長の**接尾語** $\hat{\alpha} \in (2^{\mathcal{I}})^\omega$ 及び $\hat{\beta} \in (2^{\mathcal{O}})^\omega$ に分割で

きる．そこで，有限入力列 $\bar{\alpha} = a_0 \cdot a_1 \cdots a_n \in (2^{\mathcal{I}})^*$ に対する，リアクティブシステム r の有限出力列 $r(a_0) \cdot r(a_0 \cdot a_1) \cdots r(a_0 \cdot a_1 \cdots a_n) \in (2^{\mathcal{O}})^*$ を $r^*(\bar{\alpha})$ と記す．

2.2 動作仕様

仕様とは，システムがすべきこと（あるいは逆に，すべきでないこと）を規定するものである．形式的には，**時相論理**（temporal logic）[27] [9] や **ω-オートマトン**（ω-automaton）[5] [28] [37] [17] などで与えられる．それらは振舞いを充足するもの（あるいは受理されるもの）とそれ以外に分ける．通常は，充足するもの（あるいは受理されるもの）をシステムに望まれる振舞いとみなされる．本稿では，リアクティブシステム仕様を次のように定義する．

定義 3 (仕様) 仕様は，システムに望まれる振舞いの集合 $S \subseteq (2^{\mathcal{I} \cup \mathcal{O}})^\omega$ である． ■

2.2.1 ω-正規オートマトン

安全性や活性，反応性などの形式的検証における主要な検証性質は **ω-正規**（ω-regular）と呼ばれるクラスに属する．そのため，振舞いの集合よりも抽象度が高く，ω-正規言語を受理できる **ω-正規オートマトン**（ω-regular automaton）がしばしば仕様として利用される．

定義 4 (ω-オートマトン) ω-オートマトン $\mathcal{A}$ は，5 つ組 $\langle Q, \Sigma, \Delta, q_{init}, Acc \rangle$ である．ここで，

- Q は有限の**状態集合**，
- Σ は有限の**アルファベット**，
- $\Delta : Q \times \Sigma \to 2^Q$ は，各状態 $q_i \in Q$ において文字 $s_i \in \Sigma$ を読んだ際に遷移可能な状態の集合を返す**遷移関数**，
- $q_{init} \in Q$ は**初期状態**，
- Acc は**受理条件**を規定するための要素である． ■

なお，ω-オートマトンをリアクティブシステム仕様として利用する場合，アルファベット Σ は冪集合 $2^{\mathcal{I} \cup \mathcal{O}}$ である．

オートマトン $\mathcal{A}$ の実行は，与えられた語 $\sigma = s_0 \cdot s_1 \cdot s_2 \cdots \in \Sigma^\omega$ の文字 s_i を先頭から順に読み，現在の状態 q_i（ただし $q_0 = q_{init}$）とその文字 s_i から遷移関数 Δ が与える遷移可能な状態 q_{i+1} に遷移していくことで進行する．形式的には，σ に対する**実行** ρ は，無限長状態列 $q_0 \cdot q_1 \cdot q_2 \cdots \in Q^\omega$ で与えられる．ただし，$q_0 = q_{init}$ かつすべての $i \in \mathbb{N}$ に関して $q_{i+1} \in \Delta(q_i, s_i)$ である．ここで，任意の状態 $q \in Q$ と文字 $s \in \Sigma$ について $|\Delta(q, s)| \leq 1$ であるとき，その遷移関数及びオートマトンは**決定的**（deterministic）であるという．また，σ に対する実行の集合を $Run^{\mathcal{A}}(\sigma)$ で表す．$Run^{\mathcal{A}}(\sigma)$ は複数の要素をもち得るが，$\mathcal{A}$ が決定的ならば 1 つ以下の要素しかもたない．

実行 $\rho = q_0 \cdot q_1 \cdot q_2 \cdots \in Q^\omega$ 上に無限にしばしば出現する状態の集合を $Inf(\rho)$ とする．

$$Inf(\rho) = \{q \in Q \mid \forall i \in \mathbb{N} : \exists j \in \mathbb{N}_{>i} : q = q_j\}. \quad (1)$$

ω-正規言語オートマトン $\mathcal{A}$ は，無限にしばしば出現する状態の集合 $Inf(\rho)$ に基づいて実行 ρ の特徴を捉え，受理/不受理を判定する．

定義 5 (ω-正規受理条件) *Büchi* **オートマトン** [5] の場合，Acc は受理状態集合 $F \subseteq Q$ であり，以下の *Büchi* **条件**を満たす実行 $\rho \in Run^{\mathcal{A}}(\sigma)$ が存在する語 σ を受理する．

$$Inf(\rho) \cap F \neq \emptyset. \quad (2)$$

Rabin **オートマトン** [28] の場合，Acc は受理状態の対の集合 $\mathcal{F} \subseteq 2^{Q \times Q}$ であり，以下の *Rabin* **条件**を満たす実行 $\rho \in Run^{\mathcal{A}}(\sigma)$ が存在する語 σ を受理する．

$$\bigvee_{\langle U_i, V_i \rangle \in \mathcal{F}} (Inf(\rho) \cap U_i \neq \emptyset \wedge Inf(\rho) \cap V_i = \emptyset). \quad (3)$$

■

Büchi オートマトンのうち，受理状態集合 F が状態集合全体 Q と等しいものを *safety* **オートマトン**と呼ぶ．また，受理状態集合 F が任意の文字に対して自己ループする状態，即ち，以下を満たす状態 q_f の単集合からなるものを *liveness* **オートマトン**と呼ぶ．

$$\forall s \in \Sigma : \Delta(q_f, s) = \{q_f\}. \quad (4)$$

$\mathcal{A}$ が受理する語の集合を $\mathcal{A}$ の**受理言語**といい，$L(\mathcal{A})$ で記す．$L(\mathcal{A})$ が空集合 $\emptyset$ であるか否かを判定することを**空判定**，$L(\mathcal{A})$ が語全体 Σ^ω に等しいか否

かを判定することを**全受理判定**と呼ぶ．

以下，オートマトンの種別，Büchi/Rabin/safety/liveness を B/R/S/L で表し，遷移の決定性/非決定性をそれぞれ D/N で表す．つまり，例えば，非決定性 safety オートマトンを NSA，決定性 Rabin オートマトンを DRA と表記する．

本稿では，仕様 S は S を受理する NBA として与えられている ω-正規言語と仮定する．任意の**線形時相論理**（linear temporal logic; LTL）[27] [9] 式はそれを満たす語の集合を受理言語とする指数オーダーサイズのオートマトンに変換できる [36] [15] [16] [7] [3] [24] ため，仕様が LTL 式で与えられている場合の判定手続きは本稿で与える判定手続きにオートマトン変換工程を加えるだけで容易に拡張可能である．

2.2.2 ω-ゲーム

リアクティブシステムと環境の相互作用は，2 人無限展開型ゲーム [17] としてモデル化される．1 より，先手プレイヤは環境，後手プレイヤはリアクティブシステムに対応し，交互にイベントを選択することでゲームが進行し，仕様を満たすときに後手プレイヤが勝利する．

定義 6 (ω-ゲーム) ω-ゲーム $\mathcal{G}$ は，8 つ組 $\langle Q_1, Q_2, \Sigma_1, \Sigma_2, \Delta_1, \Delta_2, q_{init}, Acc\rangle$ である．ここで，

- Q_1 は先手プレイヤが遷移先を決定する有限の**状態集合**，
- Q_2 は後手プレイヤが遷移先を決定する有限の**状態集合**，
- Σ_1 は先手プレイヤが選択するイベントの有限の**アルファベット**，
- Σ_2 は後手プレイヤが選択するイベントの有限の**アルファベット**，
- $\Delta_1 : Q_1 \times \Sigma_1 \to 2^{Q_2}$ は，先手プレイヤの各状態 $q_i \in Q_1$ においてイベント $s_i \in \Sigma_1$ が選択された際に遷移可能な状態の集合を返す**遷移関数**，
- $\Delta_2 : Q_2 \times \Sigma_2 \to 2^{Q_1}$ は，後手プレイヤの各状態 $q_i \in Q_2$ においてイベント $s_i \in \Sigma_2$ が選択された際に遷移可能な状態の集合を返す**遷移関数**，
- $q_{init} \in Q_1$ は**初期状態**，
- Acc は**勝利条件**を規定するための要素である．

ただし，状態集合 Q_1 と Q_2 並びにアルファベット Σ_1 と Σ_2 はそれぞれ互いに素であり，また，遷移関数 Δ_1 及び Δ_2 は共に決定的であるものとする． ■

リアクティブシステムと環境の相互作用をモデル化する場合，先手イベント集合 Σ_1 は入力命題の冪集合 $2^{\mathcal{I}}$，後手イベント集合 Σ_2 は出力命題の冪集合 $2^{\mathcal{O}}$ である．

オートマトンの実行に相当するゲームのプレイ $Play^{\mathcal{G}}(\alpha, \beta)$ は，両プレイヤから与えられたイベント列 $\alpha = a_0 \cdot a_1 \cdots \in (\Sigma_1)^\omega$ 及び $\beta = b_0 \cdot b_1 \cdots \in (\Sigma_2)^\omega$ に対する無限状態列 $q_{\langle 0,1\rangle} \cdot q_{\langle 0,2\rangle} \cdot q_{\langle 1,1\rangle} \cdot q_{\langle 1,2\rangle} \cdots \in (Q_1 \cdot Q_2)^\omega$ で与えられる．ただし，$Q_{\langle 0,1\rangle} = q_{init}$ かつすべての $i \in \mathbb{N}$ に関して $q_{\langle i,2\rangle} \in \Delta_1(q_{\langle i,1\rangle}, a_i)$ かつ $q_{\langle i+1,1\rangle} \in \Delta_2(q_{\langle i,2\rangle}, b_i)$ である．

勝利条件 Acc は，簡略化のため，後手プレイヤのための勝利条件とする．これは ω-オートマトンと同様に与えられ，本稿では定義 5 の条件を用いる．プレイが勝利条件を満たすとき後手プレイヤが勝利し，そうでないときには先手プレイヤが勝利する．各時点においてそれぞれのプレイヤがイベントを選択することでプレイが一意に定まり，延いては勝敗が決まる．このイベントの選択法を**戦略**と呼ぶ．

定義 7 (戦略) イベント決定関数 $\pi_1 : Q_1 \to \Sigma_1$ を先手プレイヤの戦略，$\pi_2 : Q_2 \to \Sigma_2$ を後手プレイヤの戦略と呼ぶ．また，先手プレイヤの戦略の集合を Π_1，後手プレイヤの戦略の集合を Π_2 とする．

そして，先手プレイヤの戦略 π_1 と後手プレイヤの戦略 π_2 の戦略がら誘導される次のプレイ $\rho = q_{\langle 0,1\rangle} \cdot q_{\langle 0,2\rangle} \cdot q_{\langle 1,1\rangle} \cdot q_{\langle 1,2\rangle} \cdots \in (Q_1 \cdot Q_2)^\omega$ を $Play^{\mathcal{G}}(\pi_1, \pi_2)$ とする．

$$q_{\langle 0,1\rangle} = q_{init}, \tag{5}$$

$$\forall i \in \mathbb{N} : q_{\langle i,2\rangle} \in \Delta_1(q_{\langle i,1\rangle}, \pi_1(q_{\langle i,1\rangle})), \tag{6}$$

$$\forall i \in \mathbb{N} : q_{\langle i+1,1\rangle} \in \Delta_2(q_{\langle i,2\rangle}, \pi_2(q_{\langle i,2\rangle})). \tag{7}$$

■

勝利条件が ω-正規であるときには，先手プレイヤか後手プレイヤのどちらか一方は，自身の勝利を確定付ける無記憶性の**勝利戦略**を持ち，その存在をアルゴリズム的に判定することができる [17].

定義 8 (勝利戦略) 以下を満たす戦略 π_1 を先手プレイヤの勝利戦略と呼ぶ．

$\forall \pi \in \Pi_2 : Play^{\mathcal{G}}(\pi_1, \pi)$ で先手プレイヤが勝利する． (8)

また，以下を満たす戦略 π_2 を後手プレイヤの勝利戦略と呼ぶ．

$\forall \pi \in \Pi_1 : Play^{\mathcal{G}}(\pi, \pi_2)$ で先手プレイヤが勝利する． (9)

■

2.2.3 基本操作

リアクティブシステムの実現可能性は，オートマトン/ゲーム上の操作を通して判定できる．もっとも基本的な操作は，決定化とゲーム化である．

操作 1 (決定化 [30] [20] [10] [21]) NBA $\mathcal{A}$ を DRA $\mathcal{A}'$ へ変換する操作を決定化と呼ぶ． ■

操作 2 (ゲーム化) 決定性オートマトン $\langle Q, 2^{\mathcal{I}\cup\mathcal{O}}, \Delta, q_{init}, Acc\rangle$ から以下のようなゲーム $\langle Q\cup\{q_{new1}\}, Q_2\cup\{q_{new2}\}, 2^{\mathcal{I}}, 2^{\mathcal{O}}, \Delta_1, \Delta_2, q_{init}, Acc\rangle$ を生成する操作をゲーム化と呼ぶ．ただし，Δ' は以下を満たす最小の遷移関係とする．

$$\Delta(q, a\cup b) = \{q'\} \Rightarrow \\ q_{\langle a\rangle} \in Q_2 \wedge q_{\langle a\rangle} \in \Delta_1(q, a) \wedge q' \in \Delta_2(q_{\langle a\rangle}, b), \tag{10}$$

$$\Delta(q, a\cup b) = \emptyset \Rightarrow \\ q_{\langle a\rangle} \in Q_2 \wedge q_{\langle a\rangle} \in \Delta_1(q, a) \wedge q_{new1} \in \Delta_2(q_{\langle a\rangle}, b), \tag{11}$$

$$\forall a \in 2^{\mathcal{I}} : q_{new2} \in \Delta_1(q_{new1}, a), \tag{12}$$

$$\forall b \in 2^{\mathcal{O}} : q_{new1} \in \Delta_1(q_{new2}, b). \tag{13}$$

■

なお，ゲーム化に関しては，可逆かつ見かけだけの変換であり，実装上は明示的に行う必要はない．

2.3 実現可能性

リアクティブシステムは，任意の入力列に対して，仕様を満たす出力を各ステップ毎に決定できなければならない．逆に言えば，リアクティブシステム仕様は，**充足可能**なだけでなく，**実現可能** [26] [1] [29] でなければならない．

定義 9 (充足可能) 仕様 S が充足可能ならば，かつそのときに限り，

$$\exists \alpha \in (2^{\mathcal{I}})^{\omega}, \beta \in (2^{\mathcal{O}})^{\omega} : \\ [\alpha, \beta] \in S, \tag{14}$$

すなわち，$S \neq \emptyset$. ■

手続き 1 (充足可能性判定)

入力：仕様 S を受理言語とする NBA $\mathcal{A}$

出力：「充足可能」又は「充足不能」

1. $\mathcal{A}$ の空判定を行い，受理言語が非空ならば「充足可能」，空ならば「充足不能」と判定する．

■

定義 10 (実現可能性 [26] [1] [29]) 仕様 S が実現可能ならば，かつそのときに限り，

$$\exists r \in \mathcal{R} : \\ \forall \alpha \in (2^{\mathcal{I}})^{\omega} : \\ [\alpha, r^{\omega}(\alpha)] \in S. \tag{15}$$

■

与えられた ω-正規な仕様に対する実現可能判定は**構成による証明**（proof by construction）の形態で決定可能であり，実現可能ならば，それを実現するリアクティブシステムをアルゴリズム的に生成できる [26] [22] [31] [13] [8] [14] [4]．仕様を LTL 式で与えている場合，実現リアクティブシステムを生成することを *LTL* **合成**（LTL synthesis）と呼ぶ．また，実現不能ならば，仕様違反を強制する環境（入力列生成戦略）を構成できる．

判定手法にはいくつかのアプローチがあるが，本稿では素朴なゲーム帰着アプローチを紹介する．

手続き 2 (実現可能性判定)

入力：仕様 S を受理言語とする NBA $\mathcal{A}$

出力：「実現可能」又は「実現不能」

1. $\mathcal{A}$ を DRA $\mathcal{A}'$ に決定化（操作 1）する．
2. $\mathcal{A}'$ を RG $\mathcal{G}$ にゲーム化（操作 2）する．
3. $\mathcal{G}$ のリアクティブシステム側（後手）プレイヤの勝利戦略存在判定を行い，勝利戦略 π_2 が存在するならば「実現可能」，存在しないならば「実現不能」と判定する． ■

$\mathcal{G} = \langle Q_1, Q_2, 2^{\mathcal{I}}, 2^{\mathcal{O}}, \Delta_1, \Delta_2, q_{init}, Acc\rangle$ の勝利戦略 π_2 に対して，仕様 S を実現するリアクティブシステム r は以下のように構成できる．

$$r(a_0 \cdots a_n) = \pi_2(q_{\langle n,2\rangle}) \tag{16}$$

ただし，

$$q_{\langle 0,1\rangle} = q_{init}, \tag{17}$$

$$\forall i \in \mathbb{N}_{\leq n} : q_{\langle i,2\rangle} \in \Delta_1(q_{\langle i,1\rangle}, a_i), \tag{18}$$

$$\forall i \in \mathbb{N}_{\leq n-1} : q_{\langle i+1,1\rangle} \in \Delta_2(q_{\langle i,2\rangle}, \pi_2(q_{\langle i,2\rangle})). \tag{19}$$

2.4 実現可能性必要条件

仕様が実現不能であった場合，その欠陥原因を抽出及び分析し，修正しなければならない．森ら [25] [43] や冨田ら [45] は，仕様の欠陥を分類できるようにいくつかの実現可能性必要条件（定義 11,14,16-18）を提案した．それらの実現可能性必要条件は，入力接頭語/入力接尾語/出力接頭語/出力接尾語のそれぞれに対する限量の強さ及び順序によって特徴付けられたものであった．そのため，冨田ら [34] は限量の強さ及び順序を**限量行列**（quantification matrix）として表現しその組み合わせを網羅することで実現可能性必要条件を細分化 (定義 11-20) 及び階層定理（図 1）を示した．

定義 11 (接頭的半強充足可能性 [45]) 仕様 S が接頭的半強充足可能ならば，かつそのときに限り，

$$\begin{aligned}&\forall \bar{\alpha} \in (2^{\mathcal{I}})^* : \\ &\quad \exists \hat{\alpha} \in (2^{\mathcal{I}})^\omega, \beta \in (2^{\mathcal{O}})^\omega : \\ &\qquad [\bar{\alpha} \cdot \hat{\alpha}, \beta] \in S.\end{aligned} \tag{20}$$

■

定義 12 (接尾的半強充足可能性 [34]) 仕様 S が接尾的半強充足可能ならば，かつそのときに限り，

$$\begin{aligned}&\forall \hat{\alpha} \in (2^{\mathcal{I}})^\omega : \\ &\quad \exists \bar{\alpha} \in (2^{\mathcal{I}})^*, \beta \in (2^{\mathcal{O}})^\omega : \\ &\qquad [\bar{\alpha} \cdot \hat{\alpha}, \beta] \in S.\end{aligned} \tag{21}$$

■

定義 13 (接尾的半真強充足可能性 [34]) 仕様 S が接尾的半真強充足可能ならば，かつそのときに限り，

$$\begin{aligned}&\exists \bar{\alpha} \in (2^{\mathcal{I}})^*, \bar{\beta} \in (2^{\mathcal{O}})^{|\bar{\alpha}|} : \\ &\quad \forall \hat{\alpha} \in (2^{\mathcal{I}})^\omega : \\ &\qquad \exists \hat{\beta} \in (2^{\mathcal{O}})^\omega : \\ &\qquad\quad [\bar{\alpha} \cdot \hat{\alpha}, \bar{\beta} \cdot \hat{\beta}] \in S.\end{aligned} \tag{22}$$

■

定義 14 (強充足可能性 [25] [43]) 仕様 S が強充足可能ならば，かつそのときに限り，

$$\begin{aligned}&\forall \alpha \in (2^{\mathcal{I}})^\omega : \\ &\quad \exists \beta \in (2^{\mathcal{O}})^\omega : \\ &\qquad [\alpha, \beta] \in S.\end{aligned} \tag{23}$$

■

定義 15 (真強充足可能性 [34]) 仕様 S が真強充足可能ならば，かつそのときに限り，

$$\begin{aligned}&\forall \bar{\alpha} \in (2^{\mathcal{I}})^* : \\ &\quad \exists \bar{\beta} \in (2^{\mathcal{O}})^{|\bar{\alpha}|} : \\ &\qquad \forall \hat{\alpha} \in (2^{\mathcal{I}})^\omega : \\ &\qquad\quad \exists \hat{\beta} \in (2^{\mathcal{O}})^\omega : \\ &\qquad\qquad [\bar{\alpha} \cdot \hat{\alpha}, \bar{\beta} \cdot \hat{\beta}] \in S.\end{aligned} \tag{24}$$

■

定義 16 (保存的充足可能性 [25] [43] [38]) 仕様 S が保存的充足可能ならば，かつそのときに限り，

$$\begin{aligned}&\exists r \in \mathcal{R} : \\ &\quad \forall \bar{\alpha} \in (2^{\mathcal{I}})^* : \\ &\qquad \exists \hat{\alpha} \in (2^{\mathcal{I}})^\omega, \hat{\beta} \in (2^{\mathcal{O}})^\omega : \\ &\qquad\quad [\bar{\alpha} \cdot \hat{\alpha}, r^*(\bar{\alpha}) \cdot \hat{\beta}] \in S.\end{aligned} \tag{25}$$

■

定義 17 (保存的強充足可能性 [25] [43] [38]) 仕様 S が保存的強充足可能ならば，かつそのときに限り，

$$\begin{aligned}&\exists r \in \mathcal{R} : \\ &\quad \forall \bar{\alpha} \in (2^{\mathcal{I}})^*, \hat{\alpha} \in (2^{\mathcal{I}})^\omega : \\ &\qquad \exists \hat{\beta} \in (2^{\mathcal{O}})^\omega : \\ &\qquad\quad [\bar{\alpha} \cdot \hat{\alpha}, r^*(\bar{\alpha}) \cdot \hat{\beta}] \in S.\end{aligned} \tag{26}$$

■

定義 18 (真保存的充足可能性 [45]) 仕様 S が真保存的充足可能ならば，かつそのときに限り，

$$\exists r \in \mathcal{R}:$$
$$\forall \bar{\alpha} \in (2^{\mathcal{I}})^*:$$
$$\exists \hat{\alpha} \in (2^{\mathcal{I}})^\omega:$$
$$[\bar{\alpha} \cdot \hat{\alpha}, r^\omega(\bar{\alpha} \cdot \hat{\alpha})] \in S. \quad (27)$$

■

定義 19 (安定的充足可能性 [34]) 仕様 S が安定的充足可能ならば，かつそのときに限り，

$$\exists r \in \mathcal{R}, \bar{\alpha} \in (2^{\mathcal{I}})^*, \bar{\beta} \in (2^{\mathcal{O}})^{|\bar{\alpha}|}:$$
$$\forall \hat{\alpha} \in (2^{\mathcal{I}})^\omega:$$
$$[\bar{\alpha} \cdot \hat{\alpha}, \bar{\beta} \cdot r^\omega(\hat{\alpha})] \in S. \quad (28)$$

■

定義 20 (安定的強充足可能性 [34]) 仕様 S が安定的強充足可能ならば，かつそのときに限り，

$$\exists n \in \mathbb{N}:$$
$$\forall m \in \mathbb{N}_{\geq n}, \bar{\alpha} \in (2^{\mathcal{I}})^m:$$
$$\exists \bar{\beta} \in (2^{\mathcal{O}})^m, r \in \mathcal{R}:$$
$$\forall \hat{\alpha} \in (2^{\mathcal{I}})^\omega:$$
$$[\bar{\alpha} \cdot \hat{\alpha}, \bar{\beta} \cdot r^\omega(\hat{\alpha})] \in S. \quad (29)$$

■

また，実現可能な仕様が持つべきの 4 つの特性を限量行列から説明している．

- **網羅性**（exhaustivity）：入力の接頭語及び接尾語が $\forall$ 限量であること．
- **戦略化可能性**（strategizability）：出力の接頭語及び接尾語が r 限量であること
- **保存性**（preservability）：入力の接頭語が $\forall$ 限量，出力の接頭語が r 限量であること
- **安定性**（stability）：入力の接尾語が $\forall$ 限量，出力の接尾語が r 限量であること

なお，2.1 節で述べたリアクティブシステムの特性から，通常は，接頭構造と接尾構造の非対称性，入力と出力の非対称性に基づいた以下の限量順序に従う．

- 入力の接頭語に対する限量は入力の接尾語の限量に先立つ．
- 出力の接頭語に対する限量は出力の接尾語の $\exists$ 限量に先立つ．
- 入力の接頭語/接尾語に対する $\forall$ 限量は，出力の接頭語/接尾語に対する $\exists$ 限量に先立つ．
- 出力の接頭語/接尾語に対する r 限量は，入力の接頭語/接尾語に対する $\forall$ 限量に先立つ．

ただし，接尾的半強充足可能性に関しては，接頭構造と接尾構造の非対称性に基づいた限量順序に従っていない．

3 実現可能性必要条件判定手続き

2.4 節のリアクティブシステム仕様の実現可能性必要条件群も，オートマトン/ゲーム上の操作を通して判定できる．様々な実現可能性必要条件が存在するが，網羅性，戦略化可能性，保存性，安定性の 4 つの特性 [34] を説明するものであるため，いくつかの操作の組み合わせで判定することができる．

本稿では，既知の手続きも含めてその操作の組み合わせとして再整理するとともに，これまで与えられていない必要条件にも判定手続きを与える．なお，接尾的半強充足可能性に関しては，他の実現可能性必要条件とは異なり，接頭構造と接尾構造の非対称性に基づいた限量順序に従っていないため，他の実現可能性必要条件と類似の手続きでは判定できない．

3.1 操作

各項で実現可能性必要条件判定で共通的に現れる遷移操作，状態操作，受理状態操作についてそれぞれ説明する．

3.1.1 遷移操作

強充足可能性等の実現可能性必要条件判定に対しては，遷移操作として，**出力命題除去**が用いられる．

操作 3 (出力命題除去) オートマトン $\langle Q, 2^{\mathcal{I}\cup\mathcal{O}}, \Delta, q_{init}, Acc\rangle$ を非決定性オートマトン $\langle Q, 2^{\mathcal{I}}, \Delta', q_{init}, Acc\rangle$ を生成する．ただし，Δ' は以下を満たす最小の遷移関係とする．

$$q' \in \Delta(q, a \cup b) \Rightarrow q' \in \Delta'(q, a). \quad (30)$$

■

この操作により，オートマトン上には「入力命題に対して何かしらの応答が可能である」という情報のみ抽出できる．なお，元のオートマトンが出力命題の差異

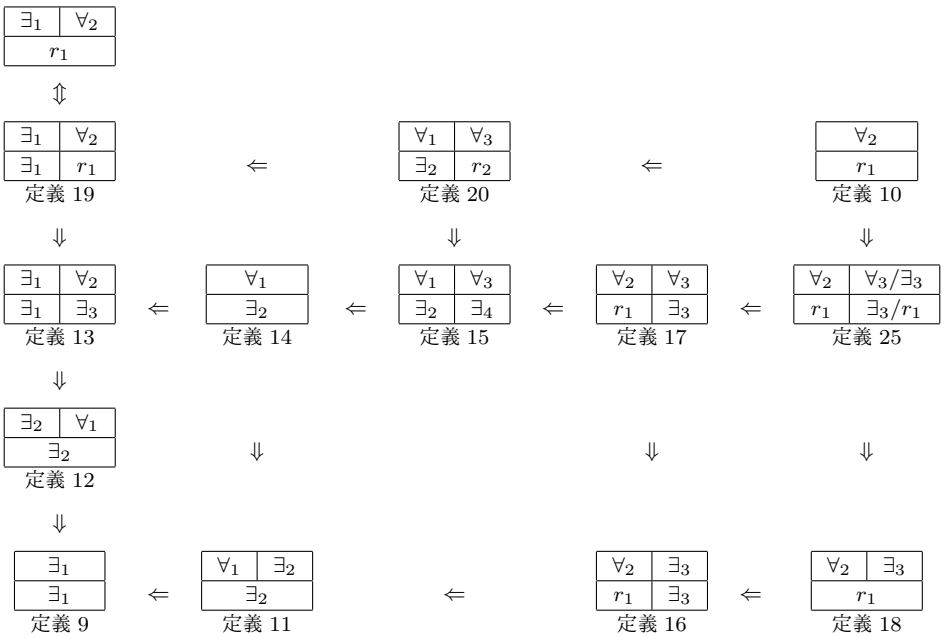

図 1　実現可能性必要条件（限量行列表現）の階層

によって遷移先が決定的であった場合には，出力命題を除去することでその決定性を失う．そのため，出力命題除去後のオートマトンは非決定的であると想定できる．

3.1.2　状態操作

オートマトン/ゲーム上には，受理/勝利に強い関連を持つ状態あるいは状態群が存在する．

定義 21 (有望状態) 決定性オートマトン $\langle Q, 2^{\mathcal{I}\cup\mathcal{O}}, \Delta, q_{init}, Acc\rangle$ において，以下の 2 条件を同時に満たす状態 $q' \in Q$ を有望状態と呼ぶ．

- q' が初期状態 q_{init} でない場合，q_{init} から q' へ到達可能である．即ち，

$$q' \in \bigcup_{i\geq 0} R_i(q_{init}), \tag{31}$$

$$R_0(q) = \{q\}, \tag{32}$$

$$R_{i+1}(q) = \{q \in \Delta(q'', s) \mid q'' \in R_i(q), s \in 2^{\mathcal{I}\cup\mathcal{O}}\}. \tag{33}$$

- $\langle Q, 2^{\mathcal{I}\cup\mathcal{O}}, \Delta, q', Acc\rangle$ の受理言語が空でない．■

定義 22 (行止状態) 決定性オートマトン $\langle Q, 2^{\mathcal{I}\cup\mathcal{O}}, \Delta, q_{init}, Acc\rangle$ において，以下を満たす状態 $q' \in Q$ を行止状態と呼ぶ．

$$\exists a \in 2^{\mathcal{I}} : \forall b \in 2^{\mathcal{O}} : \Delta(q', a \cup b) = \emptyset. \tag{34}$$

定義 23 (弱行止状態) 決定性オートマトン $\langle Q, 2^{\mathcal{I}\cup\mathcal{O}}, \Delta, q_{init}, Acc\rangle$ において，以下を満たす状態 $q' \in Q$ を弱行止状態と呼ぶ．

- $\langle Q, 2^{\mathcal{I}\cup\mathcal{O}}, \Delta, q', Acc\rangle$ の出力命題除去（操作 3）したオートマトンが全受理でない．■

定義 24 (勝利状態) 決定性オートマトン $\langle Q, 2^{\mathcal{I}\cup\mathcal{O}}, \Delta, q_{init}, Acc\rangle$ において，以下を満たす状態 $q' \in Q$ を勝利状態と呼ぶ．

- $\langle Q, 2^{\mathcal{I}\cup\mathcal{O}}, \Delta, q', Acc\rangle$ をゲーム化した場合に後手プレイヤが勝利戦略を持つ．■

保存的充足可能性等の実現可能性必要条件判定に対しては，これらの状態（あるいは状態群）に対して，次のような操作が用いられる．

操作 4 (除去) オートマトン $\langle Q, \Sigma, \Delta, q_{init}, Acc\rangle$ と指定した状態群 $Q' \subseteq Q$ からオートマトン $\langle Q \setminus Q', \Sigma, \Delta', q_{init}, Acc\rangle$ を生成する．ただし，Δ' は以下を満た

す最小の遷移関係とする．

$$\forall q, q' \in Q \setminus Q' : q' \in \Delta(q, s) \Rightarrow q' \in \Delta'(q, s). \quad (35)$$

■

操作 5 (縮退) オートマトン $\langle Q, \Sigma, \Delta, q_{init}, Acc\rangle$ と指定した状態群 $Q' \subseteq Q$ からオートマトン $\langle (Q \setminus Q') \cup \{q_{new}\}, \Sigma, \Delta', q_{init}, Acc\rangle$ を生成する．ただし，Δ' は以下を満たす最小の遷移関係とする．

$$\forall q, q' \in Q \setminus Q', s \in \Sigma :$$
$$q' \in \Delta(q, s) \Rightarrow q' \in \Delta'(q, s), \quad (36)$$
$$\forall q \in Q, q' \in Q', s \in \Sigma :$$
$$q' \in \Delta(q, s) \Rightarrow q_{new} \in \Delta'(q, s), \quad (37)$$
$$\forall s \in \Sigma : q_{new} \in \Delta'(q_{new}, s). \quad (38)$$

また，q_{new} を**縮退状態**と呼ぶ． ■

これらの操作は状態集合を変更するものであるため，一般的には間接的に受理条件要素 Acc も変更される．しかし，実現可能性必要条件判定においては，これらの状態操作の際には後述の受理条件操作が続けて行われることになる．そのため，状態操作に伴う受理条件要素 Acc の変更はここでは省略する．

3.1.3 受理条件操作

保存的充足可能性等の実現可能性必要条件判定に対しては，受理条件操作として，次のような操作が用いられる．

操作 6 (Safety 化) オートマトン $\langle Q, \Sigma, \Delta, q_{init}, Acc\rangle$ を Safety オートマトン $\langle Q, \Sigma, \Delta, q_{init}, Q\rangle$ を生成する． ■

操作 7 (Liveness 化) オートマトン $\langle Q, \Sigma, \Delta, q_{init}, Acc\rangle$ と指定した縮退状態 $q_f \subseteq Q$ から liveness オートマトン $\langle Q, \Sigma, \Delta, q_{init}, \{q_f\}\rangle$ を生成する． ■

なお，liveness 化は前述の状態縮退（操作 5）の際に追加した縮退状態 q_{new} が指定縮退状態 q_f であることを前提とする．

これら操作により，前述の受理/勝利に強い関連を持つ状態群（あるいはそれ以外の状態群）への到達性や保持性に焦点を当てたオートマトンを得ることができる．

3.2 振舞性必要条件

本節では振舞性の実現可能性必要条件群に対する判定手続きを与える．また，これまで与えられていなかった接尾的半強充足可能性，接尾的半真強充足可能性及び真強充足可能性の判定手続きについては，正当性証明も与える．なお，接尾的半強充足可能性判定手続きのみ 3.1 節の操作以外のオートマトン操作も行う．

手続き 3 (接頭的半強充足可能性判定 [45])
入力：仕様 S を受理言語とする NBA $\mathcal{A}$
出力：「接頭的半強充足可能」又は「接頭的半強充足不能」

1. $\mathcal{A}$ の非有望状態を除去（操作 4）し，
2. NSA $\mathcal{A}'$ に safety 化（操作 6）する．
3. さらに出力命題を除去（操作 3）して NSA $\mathcal{A}''$ に変換する．
4. $\mathcal{A}'$ の全受理判定を行い，全受理ならば「接頭的半強充足可能」，全受理でないならば「接頭的半強充足不能」と判定する． ■

手続き 4 (接尾的半強充足可能性判定)
入力：仕様 S を受理言語とする NBA $\mathcal{A}$
出力：「接尾的半強充足可能」又は「接尾的半強充足不能」

1. $\mathcal{A}$ を DRA $\mathcal{A}'$ に決定化（操作 3）する．
2. $\mathcal{A}'$ の非有望状態を除去（操作 4）し，
3. 出力命題も除去（操作 3）した NRA $\mathcal{A}'' = \langle Q, \Sigma, \Delta, q_{init}, \mathcal{F}\rangle$ に変換する．
4. さらに $\mathcal{A}''$ を以下を満たす NRA $\mathcal{A}''' = \langle Q', \Sigma, \Delta', q'_{init}, \mathcal{F}'\rangle$ に変換する．
   - $Q' = \{q_{\langle q_i, q_j\rangle} \mid q_i, q_j \in Q\} \cup \{q'_{init}\}$,
   - 任意の元状態 $q_i, q_j, q_k \in Q$ と文字 $s \in \Sigma$ について，
     - $q_{\langle q_i, q_j\rangle} \in \Delta'(q_{\langle q_i, q_k\rangle}, s) \Leftrightarrow q_j \in \Delta(q_k, s)$,
     - $q_{\langle q_i, q_j\rangle} \in \Delta'(q'_{init}, s) \Leftrightarrow q_j \in \Delta(q_i, s)$
   - 任意の Rabin 対 $\langle U, V\rangle \in \mathcal{F}$ について，$\langle U', V'\rangle \in \mathcal{F}' \Leftrightarrow \langle U, V\rangle \in \mathcal{F}$．ただし，
     - $q_j \in U \Leftrightarrow \forall q_i \in Q : q_{\langle q_i, q_j\rangle} \in U'$

5. $\mathcal{A}'''$ の全受理判定を行い，全受理ならば「接尾的半強充足可能」，全受理でないならば「接尾的半強充足不能」と判定する． ■

証明 1 (手続き 4 の正当性) $\mathcal{A}'''$ の構成法から，$\mathcal{A}'''$ の初期状態 q'_{init} は $\mathcal{A}''$ の全状態を模倣する．$\mathcal{A}'$ が決定的であるため，$\mathcal{A}'''$ が全受理でないならば，かつそのときに限り，その不受理語 $\hat{\alpha} \in 2^{\mathcal{I}}$ は，接尾的半強充足可能性の反例となる．すなわち，任意の入力接頭語 $\bar{\alpha} \in (2^{\mathcal{I}})^*$ と出力語 $\beta \in (2^{\mathcal{O}})^\omega$ に対して，$[\bar{\alpha} \cdot \hat{\alpha}, \beta]$ は仕様 S に含まれない． □

手続き 5 (接尾的半真強充足可能性判定)
入力：仕様 S を受理言語とする NBA $\mathcal{A}$
出力：「接尾的半真強充足可能」又は「接尾的半真強充足不能」

1. $\mathcal{A}$ を DRA $\mathcal{A}'$ に決定化（操作 3）する．
2. $\mathcal{A}'$ の非弱行止状態を縮退（操作 5）し，
3. その縮退状態を受理状態とする DLA $\mathcal{A}''$ に liveness 化（操作 7）する．
4. $\mathcal{A}''$ の空判定を行い，受理言語が非空ならば「接尾的半真強充足可能」，空ならば「接尾的半真強充足不能」と判定する． ■

証明 2 (手続き 5 の正当性) $\mathcal{A}''$ の受理言語が非空ならば，かつそのときに限り，受理状態となっている縮退状態へ到達させる有限長接頭語 $[\bar{\alpha}, \bar{\beta}] \in (2^{\mathcal{I} \cup \mathcal{O}})^*$ が存在する．$[\bar{\alpha}, \bar{\beta}]$ によって $\mathcal{A}'$ 上ではある非弱行止状態へ到達しているため，それに続く無限長接尾語として，任意の無限長入力接尾語 $\hat{\alpha} \in (2^{\mathcal{I}})^\omega$ に対して無限長出力接尾語 $\hat{\beta} \in (2^{\mathcal{O}})^\omega$ が存在し，$[\bar{\alpha} \cdot \hat{\alpha}, \bar{\beta} \cdot \hat{\beta}]$ は $\mathcal{A}'$ に受理される，即ち仕様 S に含まれる． □

手続き 6 (強充足可能性判定 [25] [43])
入力：仕様 S を受理言語とする NBA $\mathcal{A}$
出力：「強充足可能」又は「強充足不能」

1. $\mathcal{A}$ の出力命題を除去（操作 3）した NBA $\mathcal{A}'$ に変換する．
2. $\mathcal{A}'$ の全受理判定を行い，全受理ならば「強充足可能」，全受理でないならば「強充足不能」と判定する． ■

手続き 7 (真強充足可能性判定)
入力：仕様 S を受理言語とする NBA $\mathcal{A}$
出力：「真強充足可能」又は「真強充足不能」

1. $\mathcal{A}$ を DRA $\mathcal{A}'$ に決定化（操作 3）する．
2. $\mathcal{A}'$ の非弱行止状態を縮退（操作 5）し，
3. その縮退状態を受理状態とする DLA $\mathcal{A}''$ に liveness 化（操作 7）する．
4. さらに出力命題を除去（操作 3）した NLA $\mathcal{A}'''$ に変換する．
5. $\mathcal{A}'''$ の全受理判定を行い，全受理ならば「真強充足可能」，全受理でないならば「真強充足不能」と判定する． ■

証明 3 (手続き 7 の正当性) $\mathcal{A}'''$ が全受理ならば，かつそのときに限り，任意の有限長入力接頭語 $\bar{\alpha} \in (2^{\mathcal{I}})^*$ に対してある有限長出力接頭語 $\bar{\beta} \in (2^{\mathcal{O}})^*$ が存在し，この有限長接頭語 $[\bar{\alpha}, \bar{\beta}]$ によって $\mathcal{A}'$ の非弱行止状態へ到達する．到達した状態が非弱行止状態であるため，それに続く無限長接尾語として，任意の無限長入力接尾語 $\hat{\alpha}$ に対する無限長出力接尾語 $\hat{\beta}$ が存在し，$[\bar{\alpha} \cdot \hat{\alpha}, \bar{\beta} \cdot \hat{\beta}]$ は $\mathcal{A}'$ に受理される，即ち仕様 S に含まれる． □

手続き 5 における空判定は，対象オートマトンが DLA であるため，実際には受理状態への単純な到達判定である．また，手続き 6 は，初期状態の弱い行止り判定と同値である．

手続き 5 及び手続き 7 においては有限長語が接頭語として与えられた時点で弱い行止りかどうかを確定することができるように決定化が必須である．

3.3 保存性必要条件

本節では保存性の実現可能性必要条件群に対する判定手続きを与える．

手続き 8 (素朴な保存的充足可能性判定)
入力：仕様 S を受理言語とする NBA $\mathcal{A}$
出力：「保存的充足可能」又は「保存的充足不能」

1. $\mathcal{A}$ を DRA $\mathcal{A}'$ に決定化（操作 3）する．
2. $\mathcal{A}'$ の非有望状態を除去（操作 4）し，
3. DSA $\mathcal{A}''$ に safety 化（操作 6）する．

4. さらにそれを SG $\mathcal{G}$ にゲーム化（操作 2）する．
5. $\mathcal{G}$ のリアクティブシステム側（後手）プレイヤの勝利戦略存在判定を行い，勝利戦略が存在するならば「保存的充足可能」，存在しないならば「保存的充足不能」と判定する． ■

手続き 9 (保存的充足可能性判定 [25] [43] [38])
入力：仕様 S を受理言語とする NBA $\mathcal{A}$
出力：「保存的充足可能」又は「保存的充足不能」
1. $\mathcal{A}$ を DRA $\mathcal{A}'$ に決定化（操作 3）する．
2. $\mathcal{A}'$ の非有望状態を除去（操作 4）し，
3. さらに行止状態を繰り返し除去（操作 4 を反復）した DRA $\mathcal{A}''$ に変換する．
4. $\mathcal{A}''$ に初期状態が残っているならば「保存的充足可能」，残っていないならば「保存的充足不能」と判定する． ■

手続き 10 (保存的強充足可能性判定 [25] [43] [38])
入力：仕様 S を受理言語とする NBA $\mathcal{A}$
出力：「保存的強充足可能」又は「保存的強充足不能」
1. $\mathcal{A}$ を DRA $\mathcal{A}'$ に決定化（操作 3）する．
2. $\mathcal{A}'$ の非有望状態を除去（操作 4）し，
3. さらに弱行止状態を除去（操作 4）した DRA $\mathcal{A}''$ に変換する．
4. $\mathcal{A}''$ に初期状態が残っているならば「保存的強充足可能」，残っていないならば「保存的強充足不能」と判定する． ■

手続き 11 (真保存的充足可能性判定 [45])
入力：仕様 S を受理言語とする NBA $\mathcal{A}$
出力：「真保存的充足可能」又は「真保存的充足不能」
1. $\mathcal{A}$ を DRA $\mathcal{A}'$ に決定化（操作 3）する．
2. $\mathcal{A}'$ の非有望状態と行止状態を繰り返し除去（操作 4 を反復）した DRA $\mathcal{A}''$ に変換する．
3. $\mathcal{A}''$ に初期状態が残っているならば「真保存的充足可能」，残っていないならば「真保存的充足不能」と判定する． ■

手続き 9 で行われる非有望状態と行止状態を除去は，手続き 8 で構成されるゲームの非勝利状態の除去に相当する．

手続き 9 と手続き 11 の差異は，除去される状態の範囲である．手続き 9 では非有望状態を 1 度しか除去しないため，行止状態除去後に新たに発生する非有望状態が $\mathcal{A}''$ に残存している可能性がある．一方で，手続き 11 では，行止状態除去後に新たに発生する非有望状態も除去される．

3.4 安定性必要条件

本節では安定性の実現可能性必要条件群に対する判定手続き及び正当性証明を与える．

手続き 12 (安定的充足可能性判定)
入力：仕様 S を受理言語とする NBA $\mathcal{A}$
出力：「安定的充足可能」又は「安定的充足不能」
1. $\mathcal{A}$ を DRA $\mathcal{A}'$ に決定化（操作 3）する．
2. $\mathcal{A}'$ の勝利状態を縮退（操作 5）し，
3. その縮退状態を受理状態とする DLA $\mathcal{A}''$ に liveness 化（操作 7）する．
4. $\mathcal{A}''$ の空判定を行い，受理言語が非空ならば「安定的充足可能」，空ならば「安定的充足不能」と判定する． ■

証明 4 (手続き 12 の正当性) $\mathcal{A}''$ の受理言語が非空ならば，かつそのときに限り，勝利状態となっている縮退状態へ到達させる有限長接頭語 $[\bar{\alpha}, \bar{\beta}] \in (2^{\mathcal{I} \cup \mathcal{O}})^*$ が存在する．$[\bar{\alpha}, \bar{\beta}]$ によって $\mathcal{A}'$ 上ではある勝利状態へ到達しているため，それに続く任意の無限長入力接尾語 $\hat{\alpha} \in (2^{\mathcal{I}})^\omega$ に適切に応答するリアクティブシステム $r \in \mathcal{R}$ が存在し，$[\bar{\alpha} \cdot \hat{\alpha}, \bar{\beta} \cdot r^\omega(\hat{\alpha})]$ は $\mathcal{A}'$ に受理される，即ち仕様 S に含まれる． □

手続き 13 (安定的強充足可能性判定)
入力：仕様 S を受理言語とする NBA $\mathcal{A}$
出力：「安定的真強充足可能」又は「安定的真強充足不能」
1. $\mathcal{A}$ を DRA $\mathcal{A}'$ に決定化（操作 3）する．
2. $\mathcal{A}'$ の勝利状態を縮退（操作 5）し，
3. その縮退状態を受理状態とする DLA $\mathcal{A}''$ に liveness 化（操作 7）する．
4. さらに出力命題を除去（操作 3）した NLA $\mathcal{A}'''$ に変換する．

5. $\mathcal{A}'''$ の全受理判定を行い，全受理ならば「安定的真強充足可能」，全受理でないならば「安定的真強充足不能」と判定する． ■

証明 5 (手続き 13 の正当性) $\mathcal{A}'''$ が全受理ならば，かつそのときに限り，充分に長い任意の有限長入力接頭語 $\bar{\alpha} \in (2^{\mathcal{I}})^*$ に対してある有限長出力接頭語 $\bar{\beta} \in (2^{\mathcal{O}})^*$ が存在し，この有限長接頭語 $[\bar{\alpha}, \bar{\beta}]$ によって $\mathcal{A}'$ の勝利状態へ到達する．到達した状態が勝利状態であるため，それに続く任意の無限長入力接尾語 $\hat{\alpha} \in (2^{\mathcal{I}})^\omega$ に適切に応答するリアクティブシステム $r \in \mathcal{R}$ が存在し，$[\bar{\alpha} \cdot \hat{\alpha}, \bar{\beta} \cdot r^\omega(\hat{\alpha})]$ は $\mathcal{A}'$ に受理される，即ち仕様 S に含まれる． □

4 真保存的強充足可能性

保存性必要条件群の判定手続き 9-11 の対称性から，これまで知られていた保存性必要条件群より強い実現可能性必要条件が示唆される．

この必要条件は，限量行列として単純に表現できるものではないが，以下のように定義することができる．

定義 25 (真保存的強充足可能性) 仕様 S が真保存的強充足可能ならば，かつそのときに限り，

$$
\begin{aligned}
&\exists r \in \mathcal{R} : \\
&\quad \forall \bar{\alpha} \in (2^{\mathcal{I}})^* : \\
&\qquad ((\exists \hat{\alpha} \in (2^{\mathcal{I}})^\omega : \\
&\qquad\quad \langle \bar{\alpha} \cdot \hat{\alpha}, r^\omega(\bar{\alpha} \cdot \hat{\alpha}) \rangle \in S) \\
&\qquad \wedge (\forall \hat{\alpha} \in (2^{\mathcal{I}})^\omega : \\
&\qquad\quad \forall \hat{\beta} \in (2^{\mathcal{O}})^\omega : \\
&\qquad\quad \langle \bar{\alpha} \cdot \hat{\alpha}, r^\omega(\bar{\alpha}) \cdot \beta \rangle \in S)). \qquad (39)
\end{aligned}
$$

■

真保存的充足可能性では，根拠となるリアクティブシステムは都合が良い入力接尾語に対してのみ仕様を満たす応答が可能なことを保証しているに過ぎない．一方で，保存的充足可能性では，根拠となるリアクティブシステムは入力接尾語に網羅的応答できる可能性のみを維持しているに過ぎず，どのような入力語を与えても仕様を満たす応答できない可能性がある．真保存的強充足可能性では，入力接尾語に網羅的応答できる可能性を維持しつつ，都合が良い入力接尾語に対しては仕様を満たす応答が可能なことを保証できる．

判定手続きは，手続き 9 を手続き 11 に拡張するのことと同様のアプローチで，手続き 10 を拡張することで得ることができる．

手続き 14 (真保存的強充足可能性判定)
入力：仕様 S を受理言語とする NBA $\mathcal{A}$
出力：「真保存的強充足可能」又は「真保存的強充足不能」

1. $\mathcal{A}$ を DRA $\mathcal{A}'$ に決定化（操作 3）する．
2. $\mathcal{A}'$ の非有望状態と弱行止状態を繰り返し除去（操作 4 を反復）した DRA $\mathcal{A}''$ に変換する．
3. $\mathcal{A}''$ に初期状態が残っているならば「真保存的強充足可能」，残っていないならば「真保存的強充足不能」と判定する． ■

証明 6 (手続き 14 の正当性) $\mathcal{A}''$ に初期状態が残っているならば，かつその時に限り，$\mathcal{A}''$ は弱い行止りでないかつ有望な状態のみから構成されている．このとき，真保存的充足可能性の根拠となるリアクティブシステム構成法 [45] と同じ方法で真保存的強充足不能の根拠となるリアクティブシステムが構成できる．□

なお，接頭構造と接尾構造の非対称性により，このような限量の混合による実現可能性必要条件の詳細化は，安定的必要条件群には適用できない．

5 考察

5.1 計算量上界

本節では，各実現可能性必要条件判定手続きの計算量上界について述べる．

NBA の空判定及び全受理判定は，オートマトンのサイズに対してそれぞれ NLOGSPACE 完全及び PSPACE 完全である [37]．また，NBA の決定化は，オートマトンのサイズに対して EXPTIME であり，構成される DRA は指数オーダーのサイズを持つ [37]．RA から NBA への変換は，オートマトンのサイズに対して PTIME であり，構成される NRA は多項式オーダーのサイズを持つ [37]．そして，行止状態群と

勝利状態群はオートマトンのサイズに対して多項式時間で求めることができる[17].

そのため，以下の定理が成り立つ．

定理 1 (実現可能性必要条件判定の計算量上界) リアクティブシステム仕様が NBA として与えられているとき，各実現可能性必要条件判定の計算量クラスは以下の通りである．

- PSPACE : 接頭的半強充足可能性判定，強充足可能性判定．
- EXPTIME : 安定的充足可能性判定，真保存的強充足可能性判定，真保存的充足可能性判定．
- EXPSPACE : 接尾的半強充足可能性判定，接尾的半真強充足可能性判定，真強充足可能性判定，保存的強充足可能性判定，真保存的強充足可能性判定，安定的強充足可能性判定．

また，リアクティブシステム仕様が LTL 式として与えられているとき，各実現可能性必要条件判定の計算量クラスは以下の通りである．

- EXPSPACE : 接頭的半強充足可能性判定，強充足可能性判定．
- 2EXPTIME : 安定的充足可能性判定，真保存的強充足可能性判定，真保存的充足可能性判定．
- 2EXPSPACE : 接尾的半強充足可能性判定，接尾的半真強充足可能性判定，真強充足可能性判定，保存的強充足可能性判定，真保存的強充足可能性判定，安定的強充足可能性判定．

証明 7 (定理 1 の証明) （1）リアクティブシステム仕様が NBA のとき：手続き中に弱行止状態を求める必要がない場合は，手続きの構成から明らか．手続き中に弱行止状態群を求める必要がある場合，まずオートマトンの決定化が先立つ．そしてオートマトンのサイズに対して高々多項式オーダーだけ全受理判定（PSPACE 完全）を繰り返す．そのため，そのような手続きの計算量は EXPSPACE に属す．

（2）リアクティブシステム仕様が LTL 式のとき：LTL 式からオートマトンへの変換の計算量及び構成される NBA の状態数，リアクティブシステム仕様が NBA のときの計算量から明らか． □

実現可能性判定の計算量は NBA 仕様に対しては EXPTIME 完全，LTL 式仕様に対しては 2EXPTIME 完全である[26][29]．実現可能性判定より簡単な（あるいは同等の）計算量クラスとなるのは，接頭的半強充足可能性判定と強充足可能性判定しかなく，5 つの実現可能性必要条件に至っては実現可能性判定よりも難しい（あるいは同等の）計算量クラスに属する．

そのため，実現可能性必要条件判定は，高速に実現不能性を確認する手段としては有効とは言い難い．しかし，多くの手続きに共通操作があることから，実現可能性必要条件判定を並行して実施することで，時間的に効率的な欠陥分類可能ではある．ただし，どの実現可能性必要条件判定も空間計算量が大きいため，実行可能性は高いとは言えない．

実現可能性判定や自動合成技術を広く利用できるようにするためには，実現可能性及びその必要条件（あるいは充分条件）の判定を効率的に行うことが可能であり，かつ，充分な表現力を持った形式仕様記述体系が必要であろう．

5.2 接尾的半強充足可能性判定

接尾的半強充足可能性は他の実現可能性必要条件とは異なり，接頭構造と接尾構造の非対称性に基づいた限量順序に従っていない．この特徴により，他の実現可能性必要条件と類似の手続きでは判定できない．

一方で，接尾的半強充足可能性は，接頭的強充足可能性や強充足可能と同様に，これを満たさない仕様が持つ特徴を直感的な説明が可能であり，仕様が持つ欠陥を明確に指摘できる．

- 接尾的半強充足可能性の反例 :（すべての接頭語及び出力接尾語に有効な）強い悪性接尾入力語の存在
- 接頭的強充足可能性の反例 :（すべての接尾語及び出力接頭語に有効な）悪性接頭入力語の存在
- 強充足可能性の反例 :（すべての出力語に有効な）悪性入力語の存在

これは safety property[2][23] の反例が，悪性接頭語（bad prefix）であることに似ている．それに対し，接尾的半真強充足可能性は，前述の 3 つの必要条件と

は異なり，これを満たさない仕様が持つ特徴は相対的に説明しづらく，仕様が持つ欠陥を理解しづらい．

- 接尾的半真強充足可能性の反例：（各接頭語に対して個別に有効な）弱い悪性接尾入力語群の存在

そのため，接尾的半強充足可能性は，接頭構造/接尾構造の非対称性及び入力/出力の非対称性に基づいた限量順序に従わないものの，リアクティブシステム仕様の欠陥分類に有用であると考えられる．

6 関連研究

6.1 仕様分析

時間論理等で与えられる動作仕様の分析方法としては，**安全性-活性分解** [2] [23]（safety-liveness decomposition）が代表的である．仕様を安全性（safety property）と活性（liveness property）に分解することで，仕様の特性を分析できるようにするだけでなく，オートマトンへの変換や合成等をし易くであきる．

リアクティブシステム仕様のための仕様分析法としては，まず森ら [25] [43] [38] が保存的充足可能性，強充足可能性，保存的強充足可能性の 3 つの実現可能性必要条件に基づいた分析を提案している．これらの実現可能性必要条件判定の効率化については，安藤ら [40] の分散計算による効率的な安定充足可能性判定手続きや，島川ら [32] [33] は，SAT ベースの有界強充足可能性判定手続きがある．また，萩原ら [44] [18] は，仕様の欠陥箇所を特定する手法として，強充足不能の原因となる最小な部分仕様抽出する手法を提案している．

一方で，本稿では，より広範な実現可能性必要条件群に対して系統的な判定手続きを与えた．

6.2 リアクティブシステム合成

素朴な実現可能性判定手続き（手続き 2）では，ステップ 1 において，LTL2dstar[†3] [20] で実装されている Safra の構成法 [30] や，Rabinizer3[†4] で実装されている Esparza-Křetínský 法 [10] などを用いた ω-正規オートマトンの決定化が必要である．これらの手法は非常に煩雑で効率化が困難なため，実現可能性判定/自動合成では Safraless な漸進的有界合成手続き [22] [31] [13] がいくつも提案されており，Unbeast[†5] [8]，Acacia+[†6] [4]，[42] などの合成器が開発された．これらの合成器では，仕様を下方近似した safety property を漸進的に元の仕様に近づけでリアクティブシステムを合成する．なお，これらの漸進的有界近似手法群は仕様の実現可能性に関して保守的である．本稿では，実現可能性そのものではなくその必要条件を取り扱っているが，そのような保守的な漸進的近似に基づく判定は提案しておらず，改善の余地があると言える．

また，ゲームベースの実現可能性判定及び合成の手法とは異なり，SAT/SMT 等の制約ソルバーを利用した合成器として BoSy[†7] [12] [11] も開発されている．本稿で与えた実現可能性必要条件判定も制約ソルバーを利活用することで効率的に判定できる可能性がある．

LTL 合成の場合には，まず LTL 式をオートマトンに変換する必要がある [36] [15] [16] [7] [3] [24]．変換ツールとしては，SPOT[†8] [7]，LTL3BA[†9] [3]，Trans[†10] [24] などが開発されている．SPOT や LTL3BA では，オートマトンをよりコンパクトに構成できることが見込める遷移ベース受理条件をサポートしている．ただし，そのような仕組みをもっていたとしても必ずしも小さいオートマトンを構成できるとは限らない．本稿で与えた判定手続き群で操作するオートマトンは状態ベース受理条件であるが，遷移ベース受理条件にも容易に拡張できる．

6.3 不完全合成

実現可能性必要条件のうち，保存性必要条件及び保存性必要条件はある種の不完全リアクティブシステム

†3 https://www.ltl2dstar.de

†4 https://www7.in.tum.de/~kretinsk/rabinizer3.html

†5 https://www.react.uni-saarland.de/tools/unbeast/

†6 http://lit2.ulb.ac.be/acaciaplus/

†7 https://www.react.uni-saarland.de/tools/bosy/

†8 https://spot.lrde.epita.fr

†9 https://sourceforge.net/projects/ltl3ba/

†10 https://sites.google.com/site/yonezakilab/tools

の存在を保証するものである．吉浦ら [39] は保存的充足可能性，冨田ら [45] は真保存的充足可能性に基づく不完全リアクティブシステム合成手法を与えている．本稿で新たに導入した真保存的強充足可能性については，それに基づく不完全リアクティブシステム合成手法を [45] の手法を利用して行うことができる．また，Damm ら [6] は，強充足可能性の反例となる悪性入力列を許容するような不完全リアクティブシステム合成法を提案している．さらに [45] では，接頭的半強充足可能の反例となる悪性接頭入力列を許容するような不完全リアクティブシステム合成法等も提案している．

そして，平均利得に基づいて不完全な中でも最適なリアクティブシステムを合成する手法群 [46] [41] [19] [35] も提案されている．これらの手法群では，振舞い群を平均利得によって定量的に優先度付けることで，実現不能な中でも許容されるより望ましい振舞い群を抽象的に特徴付け，リアクティブシステム合成を行う．

7 おわりに

本稿では，仕様欠陥分析を実施可能にするために，実現可能性必要条件群に対する判定手続き群をいくつかの共通的操作の組合せとして与えた．また，共通的操作の組み合わせとして再整理した結果として発見された手続き間の対称性を基に，新しい実現可能性必要条件である真保存的強充足可能性及びその判定手続きを与えた．そして判定問題に対する計算量上界を示した．

リアクティブシステム仕様の欠陥分析に効果的に行うには，示唆性に富んだ検証を効率的に実施可能である必要がある．しかしながら，実現可能性自体が判定困難であるため，詳細な仕様欠陥分析には困難な必要条件（あるいは充分条件）判定が要求されてしまう．不完全合成手法群はそうした困難な問題をある程度隠蔽して「不完全であってもより望ましい」リアクティブシステムを試みるが，望ましさを追求するほど計算困難になってしまう．実現可能性判定や自動合成技術を広く利用できるようにするためには，検証/合成の効率性と仕様記述言語の充分な表現力を両立させることが可能な技術的打開策が必要であろう．

参 考 文 献

[1] Abadi, M., Lamport, L., and Wolper, P.: Realizable and Unrealizable Specifications of Reactive Systems, *Automata, Languages and Programming*, Ausiello, G., Dezani-Ciancaglini, M., and Della Rocca, S.(eds.), Lecture Notes in Computer Science, Vol. 372, Springer Berlin Heidelberg, 1989, pp. 1–17.

[2] Alpern, B. and Schneider, F. B.: Recognizing safety and liveness, *Distributed Computing*, Vol. 2, No. 3(1987), pp. 117–126.

[3] Babiak, T., Křetínský, M., Řehák, V., and Strejček, J.: LTL to Büchi Automata Translation: Fast and More Deterministic, *Tools and Algorithms for the Construction and Analysis of Systems*, Flanagan, C. and König, B.(eds.), Lecture Notes in Computer Science, Vol. 7214, Springer Berlin Heidelberg, 2012, pp. 95–109.

[4] Bohy, A., Bruyére, V., Filiot, E., Jin, N., and Raskin, J.-F.: Acacia+, a Tool for LTL Synthesis, *Computer Aided Verification*, Madhusudan, P. and Seshia, S.(eds.), Lecture Notes in Computer Science, Vol. 7358, Springer Berlin Heidelberg, 2012, pp. 652–657.

[5] Büchi, J. R.: Weak Second-Order Arithmetic and Finite Automata, *Mathematical Logic Quarterly*, Vol. 6, No. 1-6(1960), pp. 66–92.

[6] Damm, W. and Finkbeiner, B.: Automatic Compositional Synthesis of Distributed Systems, *FM 2014: Formal Methods*, Jones, C., Pihlajasaari, P., and Sun, J.(eds.), Lecture Notes in Computer Science, Vol. 8442, Springer International Publishing, 2014, pp. 179–193.

[7] Duret-Lutz, A. and Poitrenaud, D.: SPOT: an Extensible Model Checking Library Using Transition-Based Generalized Büchi Automata, *Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004. (MASCOTS 2004). Proceedings. The IEEE Computer Society's 12th Annual International Symposium on*, IEEE Computer Society, 2004, pp. 76–83.

[8] Ehlers, R.: Symbolic Bounded Synthesis, *Formal Methods in System Design*, Vol. 40, No. 2(2012), pp. 232–262.

[9] Emerson, E. A.: Temporal and Modal Logic, *Handbook of Theoretical Computer Science*, Vol. B, Elsevier, 1995, pp. 995–1072.

[10] Esparza, J. and Křetínský, J.: From LTL to Deterministic Automata: A Safraless Compositional Approach, *Computer Aided Verification*, Biere, A. and Bloem, R.(eds.), Lecture Notes in Computer Science, Vol. 8559, Springer International Publishing, 2014, pp. 192–208.

[11] Faymonville, P., Finkbeiner, B., Rabe, M. N., and Tentrup, L.: Encodings of Bounded Synthesis, *Tools and Algorithms for the Construction and Analysis of Systems*, Legay, A. and Margaria, T.(eds.), Springer Berlin Heidelberg, 2017, pp. 354–

370.

[12] Faymonville, P., Finkbeiner, B., and Tentrup, L.: BoSy: An Experimentation Framework for Bounded Synthesis, *Computer Aided Verification*, Majumdar, R. and Kunčak, V.(eds.), Springer International Publishing, 2017, pp. 325–332.

[13] Filiot, E., Jin, N., and Raskin, J.-F.: An Antichain Algorithm for LTL Realizability, *Computer Aided Verification*, Bouajjani, A. and Maler, O.(eds.), Lecture Notes in Computer Science, Vol. 5643, Springer Berlin Heidelberg, 2009, pp. 263–277.

[14] Finkbeiner, B. and Schewe, S.: Bounded synthesis, *International Journal on Software Tools for Technology Transfer*, Vol. 15, No. 5(2012), pp. 519–539.

[15] Gastin, P. and Oddoux, D.: Fast LTL to Büchi Automata Translation, *Computer Aided Verification*, Berry, G., Comon, H., and Finkel, A.(eds.), Lecture Notes in Computer Science, Vol. 2102, Springer Berlin Heidelberg, 2001, pp. 53–65.

[16] Giannakopoulou, D. and Lerda, F.: From States to Transitions: Improving Translation of LTL Formulae to Büchi Automata, *Formal Techniques for Networked and Distributed Systems – FORTE 2002*, Peled, D. and Vardi, M.(eds.), Lecture Notes in Computer Science, Vol. 2529, Springer Berlin Heidelberg, 2002, pp. 308–326.

[17] Grädel, E., Thomas, W., and Wilke, T.(eds.): *Automata Logics, and Infinite Games*, Lecture Notes in Computer Science, Vol. 2500, Springer Berlin Heidelberg, 2002.

[18] Hagihara, S., Egawa, N., Shimakawa, M., and Yonezaki, N.: Minimal Strongly Unsatisfiable Subsets of Reactive System Specifications, *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ASE '14, ACM, 2014, pp. 629–634.

[19] Hagihara, S., Ueno, A., Tomita, T., Shimakawa, M., and Yonezaki, N.: Simple Synthesis of Reactive Systems with Tolerance for Unexpected Environmental Behavior, *Proceedings of the 4th FME Workshop on Formal Methods in Software Engineering*, FormaliSE '16, ACM, 2016, pp. 15–21.

[20] Klein, J. and Baier, C.: Experiments with Deterministic Automata for Formulas of Linear Temporal Logic, *Theoretical Computer Science*, Vol. 363, No. 2(2006), pp. 182 – 195.

[21] Komárková, Z. and Křetínský, J.: Rabinizer 3: Safraless Translation of LTL to Small Deterministic Automata, *Automated Technology for Verification and Analysis*, Cassez, F. and Raskin, J.-F.(eds.), Lecture Notes in Computer Science, Vol. 8837, Springer International Publishing, 2014, pp. 235–241.

[22] Kupferman, O. and Vardi, M.: Safraless decision procedures, *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, 2005, pp. 531–540.

[23] Maretić, G. P.: LTL is closed under topological closure, *Information Processing Letters*, Vol. 114, No. 8(2014), pp. 408–413.

[24] Mochizuki, S., Shimakawa, M., Hagihara, S., and Yonezaki, N.: Fast Translation from LTL to Büchi Automata via Non-transition-based Automata, *Formal Methods and Software Engineering*, Merz, S. and Pang, J.(eds.), Lecture Notes in Computer Science, Vol. 8829, Springer International Publishing, 2014, pp. 364–379.

[25] Mori, R. and Yonezaki, N.: Several Realizability Concepts in Reactive Objects, *Information Modelling and Knowledge Bases IV*, Kangassalo, H., Jaakkola, H., Hori, K., and Kitahashi, T.(eds.), Frontiers in Artificial Intelligence and Applications, Vol. 16, IOS Press, 1993, pp. 407–424.

[26] Pnueli, A. and Rosner, R.: On the Synthesis of a Reactive Module, *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '89, ACM, 1989, pp. 179–190.

[27] Pnueli, A.: The Temporal Logic of Programs, *Foundations of Computer Science, 1977., 18th Annual Symposium on*, 1977, pp. 46–57.

[28] Rabin, M. O.: Decidability of second-order theories and automata on infinite trees, *Bull. Amer. Math. Soc.*, Vol. 74, No. 5(1968), pp. 1025–1029.

[29] Rosner, R.: *Modular Synthesis of Reactive Systems*, PhD Thesis, Weizmann Institute of Science, 1992.

[30] Safra, S.: On the Complexity of Omega Automata, *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, SFCS '88, IEEE Computer Society, 1988, pp. 319–327.

[31] Schewe, S. and Finkbeiner, B.: Bounded Synthesis, *Automated Technology for Verification and Analysis*, Namjoshi, K., Yoneda, T., Higashino, T., and Okamura, Y.(eds.), Lecture Notes in Computer Science, Vol. 4762, Springer Berlin Heidelberg, 2007, pp. 474–488.

[32] Shimakawa, M., Hagihara, S., and Yonezaki, N.: SAT-Based Bounded Strong Satisfiability Checking of Reactive System Specifications, *Information and Communication Technology*, Mustofa, K., Neuhold, E., Tjoa, A., Weippl, E., and You, I.(eds.), Lecture Notes in Computer Science, Vol. 7804, Springer Berlin Heidelberg, 2013, pp. 60–70.

[33] Shimakawa, M., Hagihara, S., and Yonezaki, N.: Complexity of Checking Strong Satisfiability of Reactive System Specifications, *Signal Processing and Information Technology*, Das, V. and Elkafrawy, P.(eds.), Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 117, Springer International Publishing, 2014, pp. 41–50.

[34] Tomita, T., Hagihara, S., Shimakawa, M., and Yonezaki, N.: A characterization on necessary con-

ditions of realizability for reactive system specifications, *Proceedings of the Workshop on Computation: Theory and Practice (WCTP 2018)*, ya Nishizaki, S., Numao, M., Caro, J., and Suarez, M. T.(eds.), 2019, pp. 33–42.

[35] Tomita, T., Ueno, A., Shimakawa, M., Hagihara, S., and Yonezaki, N.: Safraless LTL synthesis considering maximal realizability, *Acta Informatica*, Vol. 54, No. 7(2017), pp. 655–692.

[36] Vardi, M. Y. and Wolper, P.: Reasoning about Infinite Computations, *Information and Computation*, Vol. 115(1994), pp. 1–37.

[37] Vardi, M.: An Automata-Theoretic Approach to Linear Temporal Logic, *Logics for Concurrency*, Moller, F. and Birtwistle, G.(eds.), Lecture Notes in Computer Science, Vol. 1043, Springer Berlin Heidelberg, 1996, pp. 238–266.

[38] Yoshiura, N.: Decision Procedures for Several Properties of Reactive System Specifications, *Software Security - Theories and Systems*, Futatsugi, K., Mizoguchi, F., and Yonezaki, N.(eds.), Lecture Notes in Computer Science, Vol. 3233, Springer Berlin Heidelberg, 2004, pp. 154–173.

[39] Yoshiura, N. and Yonezaki, N.: Program synthesis for stepwise satisfiable specification of reactive system, *Principles of Software Evolution, 2000. Proceedings. International Symposium on*, 2000, pp. 58–67.

[40] 安藤崇央, 宮本佑樹, 萩原茂樹, 米崎直樹: リアクティブシステム仕様に対する段階的充足可能性判定器の分散オブジェクト技術を利用した実装, コンピュータ ソフトウェア, Vol. 28, No. 4(2011), pp. 262–281.

[41] 上野篤史, 冨田尭, 島川昌也, 萩原茂樹, 米崎直樹: 環境許容性のあるリアクティブシステム合成法, 電子情報通信学会技術研究報告, Vol. 114, No. 510(2015), pp. 7–12.

[42] 上野篤史, 望月翔平, 島川昌也, 萩原茂樹, 米崎直樹: LTL 式で記述されたリアクティブシステム仕様の高速な実現可能性判定器の実装に関する研究, 日本ソフトウェア科学会第 *30* 回大会論文集, 2013.

[43] 森亮靖, 友石正彦, 米崎直樹: 時相論理によるリアクティブシステム仕様の実現可能性に関する分類, コンピュータ ソフトウェア, Vol. 15, No. 3(1998), pp. 213–225.

[44] 萩原茂樹, 江川直毅, 島川昌也, 米崎直樹: 強充足不能なリアクティブシステム仕様における欠陥範囲の特定, ソフトウェア工学の基礎 *XX*, 日本ソフトウェア科学会 *FOSE 2013*, 2013, pp. 143–152.

[45] 冨田尭: 実現可能性の必要条件に基づいた不完全リアクティブシステム合成, 日本ソフトウェア科学会第 *32* 回大会論文集, 2015.

[46] 冨田尭, 山崎徹郎, 萩原茂樹, 米崎直樹: 確率的環境下において平均利得時間論理仕様を実現する最適マルコフ決定過程の自動合成法, 日本ソフトウェア科学会第 *30* 回大会論文集, 2013.