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

冨田 尭　萩原 茂樹　島川 昌也　米崎 直樹

外部環境と継続的に相互作用する高信頼リアクティブシステムを得るために，線形時相論理 LTL で記述された動作仕様の実現可能性を判定し，実現可能な場合にシステムを合成する手法がこれまでに研究されてきた．また，仕様が実現不能であった場合にその欠陥を検出・分類するために，実現可能性の幾つかの必要条件及びその判定手法も導入されている．しかし，LTL の表現力では，実現可能性と仕様記述者の意図を両立する適切な仕様を表現できない場合がある．また，現実的には，仕様を厳密には満たしていないシステムでもしばしば運用されている．そのため，不完全であっても一定の水準を満たすシステムが合成できれば有用である．

本研究では，より詳細な仕様欠陥分析を可能とするために，真段階的充足可能性，半強充足可能性，段階的許容可能性という 3 つの実現可能性必要条件を新たに導入する．そして，仕様がそれらの必要条件を満たすかを判定し，満たす場合に不完全なリアクティブシステムを合成する手続きを与える．

1　はじめに

リアクティブシステム合成（reactive system synthesis）とは，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**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外部環境からの入力を制御できないため，与えられた仕様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**[16][1][18]（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[15] は，充足可能性よりも強いいくつかの実現可能性必要条件及びそれらの充足判定手続きを提案している[15][23].

- **強充足可能性**[15]（strong satisfiability）
- **段階的充足可能性**[15]（stepwise satisfiability）
- **段階的強充足可能性**[15]（stepwise-strong satisfiability）

これらの必要条件充足判定は実現可能性判定よりも計算量が小さく，仕様に欠陥がある，即ち，仕様が実現不能である際に，その必要条件充足判定手続きの結果に基いた欠陥分析や欠陥箇所抽出[10] により，効率的な仕様修正が可能となる．また，Damm らも森らとは独立に別の必要条件を提案している [5].

- **許容可能性**[5]（admissibility）

Imperfect Reactive System Synthesis Based on Necessary Conditions of Realizability

Takashi Tomita, 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.

Shigeki Hagihara, Masaya Shimakawa, 東京工業大学大学院情報理工学研究科 計算工学専攻, Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology.

Naoki Yonezaki, 放送大学, Open University of Japan.

以上のような仕様の欠陥分析・検出やリアクティブシステム合成の研究は盛んに行われてきたが，既存の形式言語の表現力では現実的なシステム要求を正確に反映した仕様を記述できないことも多い．そのような場合，厳密に安全なシステムを得るためには，要求よりも強い性質を仕様として与えることが妥当である．しかし，合理的な強化によって仕様が実現不能となってしまうこともあり得る．また，現実には，不具合をもつシステムでもしばしば運用されている．そのため，仕様が実現不能であったとしても，可能な限り仕様を満たすようにふるまうシステムを合成することには大きな意義がある．ここで，既存の実現可能性必要条件のうち，段階的充足可能性や段階的強充足可能性，許容可能性は，ある種のリアクティブシステムの存在を示す性質である．そして，そのリアクティブシステムは，仕様を実現するとは限らないが，ある一定の水準を満たすシステムである．吉浦らは段階的充足可能な仕様から，充足可能性を保持するシステムを合成する手法を提案している [24]．また，Dammらは許容可能性な仕様から支配的リアクティブシステムを合成する手法を与えている [5]．これらは，与えられた仕様から，実現可能性必要条件に基づいた一定の水準を満たすシステムを合成する手法，即ち，**不完全合成**（imperfect synthesis）手法の一種である．そして，不完全合成された各部分システムを不備が露呈しないように組み合わせ，全体として安全な分散システムを構成する応用手法も研究されている [5]．

本研究の目的は，より詳細な仕様欠陥分析を可能とし，また，それに基づいて不完全合成を行う手法を提案することである．

本研究では，既存必要条件及び実現可能性の間の隔たりを埋める新たな必要条件として以下の 3 つの性質を導入する．

- **真段階的充足可能性**（properly-stepwise satisfiability）
  - 段階的充足可能性よりも強く，段階的強充足可能性とは比較不能な性質．
- **半強充足可能性**（semi-strong satisfiability）
  - 充足可能性よりも強く，強充足可能性や段階的充足可能性よりも弱い性質．
- **段階的許容可能性**（stepwise admissibility）
  - 許容可能性よりも強い性質．

真段階的充足可能性（resp. 半強充足可能性）は，段階的強充足可能性と実現可能性（resp. 強充足可能性）の差と同様の差を段階的充足可能性との間にもち，また，段階的強充足可能性と段階的充足可能性の差と同様の差を実現可能性（resp. 強充足可能性）との間にもつ性質である．また，本研究では，許容可能性と強充足可能性の間には，それらを満たす仕様の集合の共通部分が実現可能な仕様の集合と一致するという関係があることを示す．段階的許容可能性は，それと同様の関係を半強充足可能性との間にもつ性質である．そして，これらの必要条件の充足判定手続きとして，既存必要条件充足判定手続きと類似の手続きを与える．これにより，より詳細な仕様欠陥分析が可能となる．さらに，これら 3 つの充足可能性必要条件に基づいた不完全合成手続きとして，既存の（不完全）合成手法と類似の手続きを与える．これにより，仕様が実現不能であった場合でも，導入した実現可能性必要条件を満たしていれば，既存の不完全合成手法よりもより望ましい水準を満たすシステムの合成が可能となる．

本稿の構成は以下の通りである．2 章では，準備として，リアクティブシステムや線形時間論理 LTL，オートマトンの定義を与え，また，仕様の実現可能性の定義及び既存のその必要条件について紹介する．3 章では，新たに 3 つの実現可能性必要条件を導入し，その充足判定手続きを与える．4 章では，新たに導入した必要条件も含め，実現可能性必要条件の間の包含関係に関する階層定理を示す．5 章では，新たに導入した必要条件に基づいた不完全合成手法を与える．そして 6 章で関連研究について述べ，最後に 7 章でまとめる．

2 準備

本章では，リアクティブシステム，その仕様記述言語としてしばしば用いられる線形時間論理 LTL，そして，オートマトンの定義を与える．

また，仕様の実現可能性の定義及びその必要条件を紹介する．

2.1 リアクティブシステム

制御システムなどの環境（ユーザや外部モジュール）と継続的に相互作用するオープンシステムは，**リアクティブシステム**としてしばしばモデル化される．この種のシステムの特性として，以下の二点が挙げられる．

特性 1: リアクティブシステムは出力を制御できるが，外部環境からの入力は制御できない．

特性 2: リアクティブシステムは過去の相互作用履歴のみを用いて現在の出力を決定しなければならない．

形式的には，リアクティブシステムは以下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．

ただし，$\mathcal{I}/\mathcal{O}$ は外部環境/システムが制御する入力/出力イベントの生起を表す互いに素，即ち，$\mathcal{I} \cap \mathcal{O} = \emptyset$ である入力/出力原子命題集合である． ■

以下では，すべてのリアクティブシステムの集合を $\mathcal{R}$ で表す．

定義 2 (ふるまい)．外部環境からの無限長入力列 $\alpha = a_0a_1a_2\cdots \in (2^{\mathcal{I}})^\omega$ に対するリアクティブシステム r の無限長出力列 $r(a_0)r(a_0a_1)r(a_0a_1a_2)\cdots \in (2^{\mathcal{O}})^\omega$ を $r^\omega(\alpha)$ とする．

このとき，α に対する r の**ふるまい**は，各点和集合 $\alpha \sqcup r^\omega(\alpha) \in (2^{\mathcal{I}\cup\mathcal{O}})^\omega$，即ち，$(a_0 \cup r(a_0))(a_1 \cup r(a_0a_1))(a_2 \cup r(a_0a_1a_2))\cdots$ である． ■

入力/出力列を有限長前半部分列と無限長後半部分に分割したとき，それらをそれぞれ**入力/出力接頭辞**（input/output prefix）と**入力/出力接尾辞**（input/output suffix）と呼ぶ．また，入力接頭辞 $\tilde{x} = a_0a_1\cdots a_n \in (2^{\mathcal{I}})^+$ に対するリアクティブシステム r の出力接頭辞 $r(a_0)r(a_0a_1)\cdots r(a_0a_1\cdots a_n)$ を $r^*(\tilde{x})$ で表す．

2.2 線形時相論理 LTL

線形時相論理 [17]（Linear Temporal Logic，LTL）は，一つの時間系列上の性質を記述することができる時間論理である．動的システムのふるまいの性質（動作仕様）の記述にしばしば用いられ，**モデル検査**（model checking）でも広く用いられている．LTL 式は，命題論理の論理結合子 $\neg$，$\vee$，$\wedge$，$\to$，$\leftrightarrow$ と**時間演算子** $\bigcirc$，$\Diamond$，$\Box$，$\mathtt{U}$ を用いて記述される．

定義 3 (LTL の構文)．原子命題集合 $\mathcal{P}$ 上の LTL 式 φ の構文は以下の通りである．

$$\varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \to \varphi \mid \varphi \leftrightarrow \varphi \mid \bigcirc\varphi \mid \Diamond\varphi \mid \Box\varphi \mid \varphi\,\mathtt{U}\,\varphi. \tag{1}$$

ただし，$a \in \mathcal{P}$ は原子命題である． ■

なお，リアクティブシステムの仕様記述に用いる際には $\mathcal{P} = \mathcal{I} \cup \mathcal{O}$ である．

直感的には，$\bigcirc\varphi$ は「次のステップで φ を満たす」，$\Diamond\varphi$ は「いつか φ を満たす」，$\Box\varphi$ は「常に φ を満たす」，$\varphi_1\,\mathtt{U}\,\varphi_2$ は「いつか φ_2 を満たし，それまでの間は φ_1 を満たす（強い until)」ことをそれぞれ表す．

LTL では，これらの演算子を組み合わせて，例えば，以下のような性質を記述できる．

$$\Box\Diamond\varphi, \tag{2}$$

$$\Diamond\Box\varphi, \tag{3}$$

$$\Box(\varphi_1 \to \Diamond\varphi_2). \tag{4}$$

式 (2) は「無限にしばしば φ が成り立つ」，式 (3) は「ある時刻以後，常に φ が成り立つ」，式 (4) は「常に，φ_1 が成り立った後には必ず φ_2 が成り立つ」という性質をそれぞれ表す．

LTL の意味論，即ち，無限長語 $\sigma \in (2^{\mathcal{P}})^\omega$ と LTL 式 φ の間の充足関係 $\models$ は通常通りの定義に従うものとし，本稿では省略する．

LTL 式 φ で与えられた仕様を満たすふるまいが存在するとき，φ は**充足可能**（あるいは無矛盾）であるという．

定義 4 (充足可能性)．LTL 式 φ について，以下を満たすとき，φ は充足可能であるという．

$$\exists\sigma \in (2^{\mathcal{P}})^\omega : \sigma \models \varphi. \tag{5}$$

■

LTL 式 φ は，それを満たす語の集合（即ち，言語）$\{\sigma \in (2^{\mathcal{P}})^\omega \mid \sigma \models \varphi\}$ を規定する．これを $\mathcal{L}(\varphi)$ で表す．

また，LTL 式 φ に対して，任意の接尾辞と連結しても φ を満たすことがない，即ち，以下を満たす接頭辞 $s_0\cdots s_n \in (2^{\mathcal{P}})^*$ を**悪性接頭辞**（bad prefix）と呼ぶ．

$$\forall \sigma \in (2^{\mathcal{P}})^{\omega} : s_0 \cdots s_n \sigma \not\models \varphi. \tag{6}$$

φ が悪性接頭辞をもたないとき，φ は **safety property** であるという．そして，φ の非悪性接頭辞のみを接頭辞としてもつ語からなる言語，即ち，以下のように定義される言語 $Safe(\varphi)$ を φ の **safety 成分**という．

$$Safe(\varphi) = \{s_0 s_1 \cdots \in 2^{\mathcal{P}} \mid \text{任意の } i \in \mathbb{N} \text{ に関して } s_0 \cdots s_i \text{ は } \varphi \text{ の悪性接頭辞でない } \}. \tag{7}$$

2.3 オートマトン

与えられた無限長語に対して，**受理**/**不受理**を判定する有限状態機械を **ω-オートマトン**（ω-automaton）という．

2.3.1 定義

形式的には．以下のように定義される．

定義 5 (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．ただし，

- 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．■

オートマトン $\mathcal{A}$ の実行は，与えられた語 $\sigma = s_0 s_1 s_2 \cdots \in \Sigma^{\omega}$ 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 $q_0 q_1 q_2 \cdots \in Q^{\omega}$ 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．そして，無限状態列 $\rho = q_0 q_1 q_2 \cdots \in Q^{\omega}$ 上に無限にしばしば出現する状態の集合を $Inf(\rho)$ とする．

$$Inf(\rho) = \{q \in Q \mid \forall i \in \mathbb{N} : \exists j \in \mathbb{N}_{>i} : q = q_j\}. \tag{8}$$

ω-オートマトン $\mathcal{A}$ は，実行の特徴と Acc 及びオートマトンの種類に応じて受理/不受理を判定する．そのような受理語の集合を $\mathcal{A}$ の**受理言語**という，LTL 式 φ に関するモデル検査等の形式的検証は，$\mathcal{L}(\varphi)$ を受理言語とするオートマトンに対するグラフ解析により行うことができる．LTL により規定できる言語のクラスは，**ω-正規言語**（ω-regular language）のクラスの部分集合であり，**Büchi オートマトン**（BA）や **Rabin オートマトン**（RA）などの **ω-正規オートマトン**によって認識できる．この 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 6 (Büchi オートマトン)**.** BA は，ω-オートマトン $\langle Q, \Sigma, \Delta, q_{init}, F\rangle$ である．ただし，

- $F \subseteq Q$ は受理状態集合であり，
- 以下の **Büchi 受理条件**を満たす受理実行 $\rho \in Run^{\mathcal{A}}(\sigma)$ が存在する語 σ を受理する．

$$Inf(\rho) \cap F \neq \emptyset. \tag{9}$$

■

定義 7 (Rabin オートマトン)**.** RA は，ω-オートマトン $\langle Q, \Sigma, \Delta, q_{init}, \mathcal{F}\rangle$ である．ただし，

- $\mathcal{F} \subseteq 2^{Q \times Q}$ は，無限にしばしば訪れるべき集合 $U_i \subseteq Q$ と高々有限回しか訪れるべきでない集合 $V_i \subseteq Q$ の対の集合であり，
- 以下の **Rabin 受理条件**を満たす受理実行 $\rho \in Run^{\mathcal{A}}(\sigma)$ が存在する語 σ を受理する．

$$\bigvee_{\langle U_i, V_i\rangle \in \mathcal{F}} (Inf(\rho) \cap U_i \neq \emptyset \wedge Inf(\rho) \cap V_i = \emptyset). \tag{10}$$

■

決定性 RA の受理言語のクラスは ω-正規言語のクラスと一致している．一方で，決定性 BA の受理言語のクラスは，それよりも真に小さく，また，LTL により規定できる言語のクラスとは比較不能である．本稿では，決定性と明示されたオートマトンは決定性 RA を指し，そうでないオートマトンは BA を指すこととする．以降では，この 2 種のオートマトン

を利用する手続きを扱うが，これらと同等な表現能力をもつオートマトンを利用することも可能である．

2.3.2 強連結成分

オートマトン $\mathcal{A} = \langle Q, \Sigma, \Delta, q_{init}, Acc \rangle$ 上で，状態 q_0 から状態 q_n へ遷移可能であるとは，ある有限長状態列 $p = q_0 \dots q_n \in Q^+$ が存在し，すべての $i \in \mathbb{N}_{<n}$ に関して，ある文字 s_i が存在し $q_{i+1} \in \Delta(q_i, s_i)$ であることである．そのような p を（q_0 を始点とする）**パス**といい，状態 q を始点とするすべてのパスの集合を $Path^{\mathcal{A}}(q)$ で表す．そして，互いに到達可能である極大な状態集合を**強連結成分**（strongly-connected component，SCC）という．オートマトン（有向グラフ）上のすべての SCC を列挙することを **SCC 分解**（SCC decomposition）といい，Tarjan のアルゴリズム等により効率的に計算できる．

また，初期状態 q_{init} を始点とするあるパス $p_1 \in Path^{\mathcal{A}}(q_{init})$ が存在し，$p_1 p_2^\omega$ が受理実行（例えば，$\mathcal{A}$ が BA であるときには $Inf(p_2^\omega) \cap F \neq \emptyset$）となるような巡回パス p_2 を**受理閉路**と呼ぶ．そして，受理閉路を含む SCC を**受理 SCC** と呼ぶ．

2.3.3 LTL 式からオートマトンへの変換

LTL 式 φ から $\mathcal{L}(\varphi)$ を受理言語とする，即ち，等価な BA への効率的に変換する手法 [6] [2] [14] は盛んに研究されており，変換器として SPOT[†1] [6]，LTL3BA[†2] [2]，Trans[†3] [14] などが開発されている．

また，LTL 式から等価な決定性 RA への変換器としては，Safra の構成法 [19] を実装した LTL2dstar[†4] [12] や，Safraless[†5] な変換手法 [8] を実装した Rabinizer3[†6] が開発されている．

†1 https://spot.lrde.epita.fr/

†2 http://sourceforge.net/projects/ltl3ba/

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

†4 http://www.ltl2dstar.de/

†5 Esparza-Křetínský 法 [8] は，Safra の構成法を用いずに LTL 式から等価な決定性 ω-正規オートマトンを構成する手法であるため，Safraless と形容される．しかし，実現可能性判定/自動合成においては，決定性 ω-正規オートマトンではなく，決定性 safety オートマトン等のより表現力の弱いオートマトンを利用した手法/アプローチを Safraless と形容する．

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

2.4 実現可能性

リアクティブシステムの仕様 φ は充足可能（無矛盾）なだけでは不十分である．なぜならば，リアクティブシステムは，2.1 節で述べた特性 1 と特性 2 を満たしながら，任意の入力列に対して仕様を満たすようにふるまえなければならないためである．与えられた仕様に対して，そのようなリアクティブシステムが存在するとき，その仕様は**実現可能**である．

定義 8 (実現可能性)**.** 言語 $L \subseteq (2^{\mathcal{P}})^\omega$ について，以下を満たすリアクティブシステム $r \in \mathcal{R}$ が存在するとき，L は実現可能であるという．

$$\forall \alpha \in (2^{\mathcal{I}})^\omega : \alpha \sqcup r^\omega(\alpha) \in L. \tag{11}$$

また，LTL 式 φ について，$L(\varphi)$ が実現可能であるとき，即ち，以下を満たす $r \in \mathcal{R}$ が存在するとき，φ は実現可能 [16] [1] であるという．

$$\forall \alpha \in (2^{\mathcal{I}})^\omega : \alpha \sqcup r^\omega(\alpha) \models \varphi. \tag{12}$$

■

本稿ではその詳細については省略するが，素朴な実現可能性判定手続きの概略は以下の通りである．

手続き 9 (実現可能性判定手続き)**.**

入力：LTL 式 φ

出力：φ が実現可能か否か

1. $\mathcal{L}(\varphi)$ を受理する決定性オートマトン $\mathcal{A}_\varphi$ を構成する．
2. $\mathcal{A}_\varphi$ の状態及び遷移を入力/出力のターンに分割し，二人ゲーム $\mathcal{G}_\varphi$ に変換する．
   - $\mathcal{G}_\varphi$ 上では，φ を満たす振る舞いに対応するプレイでは出力側（即ち，リアクティブシステム側）プレイヤが勝利する．
3. $\mathcal{G}_\varphi$ 上の出力側プレイヤの勝利戦略が存在するならば「実現可能」と判定し，そうでないならば「実現不能」と判定する． ■

ゲーム上の戦略はリアクティブシステムと見なせるため，この勝利戦略は仕様を実現するリアクティブシステムである．よって，自動合成では，この勝利戦略を出力する．

LTL 式の実現可能性判定及び自動合成は 2EXPTIME 完全 [16] [18] であり，計算コストは非常に大きい．さまざまな効率化手法 [13] [20] [9] 及びツール [11] [7] [4] [26] が提案されているが，(1) 仕様と等価な

決定性オートマトンを構成し，(2) それに対応するゲームに変換し，(3) そのゲームの勝利戦略を求める，という流れはどの手法/ツールも同じである †7.

2.5 既存の実現可能性必要条件

本節では，森ら [15] 及び Damm ら [5] が提案した実現可能性必要条件について紹介する．

2.5.1 強充足可能性

強充足可能性は，2.1 節で述べた特性 1 のみに注目した，「任意の入力列に対して仕様を満たす出力列が存在する」という実現可能性必要条件である．

定義 10 (強充足可能性 [15])．LTL 式 φ について，以下を満たすとき，φ は強充足可能であるという．

$$\forall \alpha \in (2^{\mathcal{I}})^{\omega} : \exists \beta \in (2^{\mathcal{O}})^{\omega} : \alpha \sqcup \beta \models \varphi. \quad (13)$$

■

ここで，φ の強充足可能性の反例となる入力列，即ち，以下を満たす無限長入力列 $\alpha \in (2^{\mathcal{I}})^{\omega}$ を**悪性入力列**（bad input sequence）と呼ぶ．

$$\forall \beta \in (2^{\mathcal{O}})^{\omega} : \alpha \sqcup \beta \not\models \varphi. \quad (14)$$

φ の強充足可能性の反例となる悪性入力列は φ が成り立たないように強制する無限長入力列である．そのような入力列に対する任意の振る舞いの集合を $CST(\varphi)$ で表す．

$$CST(\varphi) = \{\alpha \sqcup \beta \in (2^{\mathcal{P}})^{\omega} \mid \beta \in (2^{\mathcal{O}})^{\omega} \text{ and } \alpha \text{ は } \varphi \text{ の悪性入力列 } \}. \quad (15)$$

2.5.2 段階的充足可能性

段階的充足可能性は，「各時刻において充足可能性を保持するような出力を決定できるリアクティブシステムが存在する」という実現可能性必要条件である．

定義 11 (段階的充足可能性 [15])．LTL 式 φ について，以下を満たすリアクティブシステム $r \in \mathcal{R}$ が存在するとき，φ は段階的充足可能であるという．

$$\forall \tilde{x} \in (2^{\mathcal{I}})^{*} : \exists \alpha \in (2^{\mathcal{I}})^{\omega} : \exists \beta \in (2^{\mathcal{O}})^{\omega} : (\tilde{x} \sqcup r^{*}(\tilde{x}))(\alpha \sqcup \beta) \models \varphi. \quad (16)$$

■

即ち，φ が段階的充足可能ならば，かつそのときに限り，φ の safety 成分 $Safe(\varphi)$ は実現可能である．

†7 交代木オートマトン及びその全受理判定を利用するものもあるが，それらはゲームとその勝利戦略探索と対応しており本質的には同じである．

2.5.3 段階的強充足可能性

段階的強充足可能性は，強充足可能性と段階的充足可能性を掛け合わせた，「各時刻において強充足可能性を保持するような出力を決定するリアクティブシステムが存在する」という実現可能性必要条件である．

定義 12 (段階的強充足可能性 [15])．LTL 式 φ について，以下を満たすリアクティブシステム $r \in \mathcal{R}$ が存在するとき，φ は段階的強充足可能であるという．

$$\forall \tilde{x} \in (2^{\mathcal{I}})^{*}, \forall \alpha \in (2^{\mathcal{I}})^{\omega} : \exists \beta \in (2^{\mathcal{O}})^{\omega} : (\tilde{x} \sqcup r^{*}(\tilde{x}))(\alpha \sqcup \beta) \models \varphi. \quad (17)$$

■

なお，強充足可能かつ段階的充足可能であることと段階的強充足可能であることは一致しない [15]．

2.5.4 許容可能性

許容可能性は，ゲーム理論の**支配**（dominance）の概念に基づいた，「仕様を満たすような出力列が存在しない入力列を除くどのような入力列に対しても仕様を満たすようにふるまうリアクティブシステムが存在する」という実現可能性必要条件である．

定義 13 (許容可能性 [5])．LTL 式 φ について，以下を満たすリアクティブシステム $r \in \mathcal{R}$ が存在するとき，φ は許容可能であるという．

$$\forall \alpha \in (2^{\mathcal{I}})^{\omega} : ((\exists \beta \in (2^{\mathcal{O}})^{\omega} : \alpha \sqcup \beta \models \varphi) \Rightarrow \alpha \sqcup r^{\omega}(\alpha) \models \varphi). \quad (18)$$

また，そのような r は φ に関して支配的であるという． ■

即ち，支配的リアクティブシステムは，すべての入力列に対して最適な出力列を応答するシステムである．

3 新たな実現可能性必要条件

本章では，2.5 節で紹介した既存実現可能性必要条件及び実現可能性の間の隔たりを埋める新たな必要条件として，真段階的充足可能性と半強充足可能性，段階的許容可能性の 3 つの必要条件を各節で導入し，その判定手法を与える．

3.1 真段階的充足可能性

本節では，新たな実現可能性必要条件として**真段階**

的充足可能性を導入し，その判定手続きを与える．

段階的充足可能性の証拠となる，即ち，式 (16) を満たすリアクティブシステム r は，任意の入力接頭辞に対して充足可能性は保持している．しかしながら，段階的充足可能性はあくまで仕様の safety 成分を実現するシステムの存在を示すものであり，式 (16) 中の出力接尾辞 β は r に束縛されていないため，r はどのような入力列に対しても φ を満たすようにふるまうことができない可能性がある．つまり，φ の safety 成分 $Safe(\varphi)$ を実現するためのいかなる戦略に従うことも，φ 自体を満たすために必要な動作を阻害してしまう可能性がある．

そこで,「各時刻において充足可能性を “真” に保持するような出力を決定できるリアクティブシステムが存在する」という実現可能性必要条件として，新たに真段階的充足可能性を定義する．

定義 14 (真段階的充足可能性)**.** LTL 式 φ について，以下を満たすリアクティブシステム $r \in \mathcal{R}$ が存在するとき，φ は真段階的充足可能であるという．

$$\forall \tilde{x} \in (2^{\mathcal{I}})^* : \exists \alpha \in (2^{\mathcal{I}})^\omega : (\tilde{x}\alpha) \sqcup r^\omega(\tilde{x}\alpha) \models \varphi. \tag{19}$$

■

真段階的充足可能性の証拠となる，即ち，式 (19) を満たすリアクティブシステム r は，任意の入力接頭辞に対して充足可能性は保持し，また，何らかの入力接尾辞に対しては仕様をみたすようにふるまうことが可能なシステムである．段階的充足可能性と段階的強充足可能性の定義から，実現可能性は真段階的強充足可能性ともいうべき性質であり，真段階的充足可能性と段階的充足可能性の差は実現可能性と段階的強充足可能性の差と類似している．

この真段階的充足可能性判定手続きは以下のように与えられる．

手続き 15 (真段階的充足可能性判定手続き)**.**

入力：LTL 式 φ

出力：φ が真段階的充足可能か否か

1. $\mathcal{L}(\varphi)$ を受理する決定性オートマトン $\mathcal{A}_\varphi$ を構成する．
2. $\mathcal{A}_\varphi$ に対して以下の 2 ステップの状態除去操作を変化がなくなるまで繰り返し，初期状態が残ったならば「真段階的充足可能」と判定し，そうでないならば「真段階的充足不能」と判定する．
   (a) 初期状態から到達不能または受理閉路（の上に現れる状態）へ到達不能な状態をすべて取り除く．
   (b) **行き止まり**状態をすべて取り除く．（行き止まり状態が存在しないように除去する）■

ここで，$\mathcal{A} = \langle Q, 2^{\mathcal{P}}, \Delta, q_{init}, Acc\rangle$ の状態 q が**行き止まり**であるとは，ある入力 $a \in 2^{\mathcal{I}}$ が存在し，任意の出力 $b \in 2^{\mathcal{O}}$ について，$\Delta(q, a \cup b) = \emptyset$，即ち，$q$ から $a \cup b$ で遷移可能な状態が存在しないことである．

段階的充足可能は，状態除去操作 (a) と (b) を 1 回ずつだけ行うことで判定できる．1 回ずつしか行わないため，(b) で行き止まり状態が除去されたことにより，受理閉路へ到達不能となってしまう状態が存在し得る．一方で，(a) と (b) を繰り返す真段階的強充足可能性判定では，行き止まる状態や，受理閉路へ繰り返し到達可能な状態は存在しない．

この手続きの正当性は，5.1 節で示す．

3.2 半強充足可能性

本節では，新たな実現可能性必要条件として**半強充足可能性**を導入し，その判定手続きを与える．

真段階的充足可能性は，任意の時刻までの有限長出力列（即ち，各時刻の出力）と段階的充足可能性がその存在を保証していた出力接尾辞に関して，一貫した生成戦略が存在することを保証した性質ともいえる．この性質は式 (16) 中の出力接尾辞 β を r で束縛することで得られたが，逆に，出力接頭辞 $r^*(\tilde{x})$ の r による束縛を排除した性質を考えることもできる．この性質は,「各時刻において，必要であれば過去の出力を改竄することで，充足可能性を保持できるリアクティブシステムが存在する」という段階的充足可能性よりも弱い性質であり，また,「任意の入力接頭辞に対して，仕様を満たすような入力接尾辞及び出力列が存在する」という強充足可能性よりも弱い性質でもある．

そこで，この新しい実現可能性必要条件を半強充足可能性と定義する．†8

†8 本文中でも述べているように，この性質は**半段階的充足可能性**（semi-stepwise satisfiability）ともいえる

定義 16 (半強充足可能性)**.** LTL 式 φ について，以下を満たすとき，φ は半強充足可能であるという．

$$\forall \tilde{x} \in (2^{\mathcal{I}})^* : \exists \alpha \in (2^{\mathcal{I}})^\omega, \exists \beta \in (2^{\mathcal{O}})^\omega : (\tilde{x}\alpha) \sqcup \beta \models \varphi. \quad (20)$$

■

ここで，φ の半強充足可能性の反例となる入力接頭辞，即ち，以下を満たす有限長入力列 $\tilde{x} \in (2^{\mathcal{I}})^*$ を**悪性入力接頭辞**（bad input prefix）と呼ぶ．

$$\forall \alpha \in (2^{\mathcal{I}})^\omega,\ \forall \beta \in (2^{\mathcal{O}})^\omega,\ (\tilde{x}\alpha) \sqcup \beta \models \varphi. \quad (21)$$

φ の半強充足可能性の反例となる悪性入力接頭辞は φ が成り立たないことを決定付ける有限長入力列である．そのような接頭辞をもつ入力列に対する任意の振る舞いの集合を $CSEST(\varphi)$ で表す．

$$CSEST(\varphi) = \{\tilde{x}\alpha \sqcup \beta \in (2^{\mathcal{P}})^\omega \mid \alpha \in (2^{\mathcal{I}})^\omega,\ \beta \in (2^{\mathcal{O}})^\omega \text{ and } \tilde{x} \text{ は } \varphi \text{ の悪性入力接頭辞 }\}. \quad (22)$$

この半強充足可能性判定手続きは以下のように与えられる．

手続き 17 (半強充足可能性判定手続き)**.**

入力：LTL 式 φ

出力：φ が半強充足可能か否か

1. $\mathcal{L}(\varphi)$ を受理するオートマトン $\mathcal{A}_\varphi$ を構成する．
2. $\mathcal{A}_\varphi$ の受理閉路に到達可能な状態をすべて受理状態に変更し，そうでない状態を取り除く．
   - 変更されたオートマトンを $\mathcal{A}_\varphi^{safe}$ とする．
3. $\mathcal{A}_\varphi^{safe}$ にラベル付けされている出力命題をドントケアに変更する．
   - 変更されたオートマトンを $\mathcal{A}_\varphi^{SeSt}$ とする．
4. $\mathcal{A}_\varphi^{SeSt}$ の補オートマトン $\overline{\mathcal{A}_\varphi^{SeSt}}$ を構成する．
5. $\overline{\mathcal{A}_\varphi^{SeSt}}$ の空判定を行い，空ならば「半強充足可能」と判定し，そうでないならば「半強充足不能」と判定する． ■

ここで，$\mathcal{A} = \langle Q, 2^{\mathcal{P}}, \Delta, q_{init}, Acc\rangle$ にラベル付けされている出力命題をドントケアに変更するとは，遷移関数 Δ を，以下を満たす遷移関数 Δ' で置き換えることを指す．

$$\forall q \in Q : \forall s \subseteq \mathcal{P} : \Delta'(q,s) = \bigcup_{b \subseteq \mathcal{O}} \Delta(q, (s \cap \mathcal{I}) \cup b). \quad (23)$$

定理 18. 手続き 17 は正当な半強充足可能性判定手続きである． ■

(**証明**) まず，$\overline{\mathcal{A}_\varphi^{SeSt}}$ が $CSEST(\varphi)$ を受理言語とするオートマトンであることを示す．ステップ 2 において構成される $\mathcal{A}_\varphi^{safe}$ は，φ の悪性接頭辞をもたない語の集合，即ち，φ の safety 成分 $Safe(\varphi)$ を受理言語とするオートマトンである．そのため，そのオートマトンの出力命題をドントケアに変更した $\mathcal{A}_\varphi^{SeSt}$ は，なんらかの出力接頭辞 $\tilde{y} \in (2^{\mathcal{O}})^*$ が存在しそれとの各点和からなる接頭辞 $\tilde{x} \sqcup \tilde{y}$ が悪性接頭辞になることがない入力接頭辞 $\tilde{x} \in (2^{\mathcal{I}})^*$ のみをもつ入力列を受理するオートマトンである．よって，$\mathcal{A}_\varphi^{SeSt}$ の補オートマトン $\overline{\mathcal{A}_\varphi^{SeSt}}$ は，$CSEST(\varphi)$ を受理言語とするオートマトンである．

また，定義 16 と式 (22) より．$\overline{\mathcal{A}_\varphi^{SeSt}}$ の受理言語が空であるならば，かつその時に限り，φ は半強充足可能である．

以上により示された． □

なお，強充足可能性は，ステップ 2 を飛ばし，ステップ 3 において「$\mathcal{A}_\varphi$ にラベル付けされている出力命題をドントケアに変更する」ことで最終的に得られる $CST(\varphi)$ を受理するオートマトン $\overline{\mathcal{A}_\varphi^{St}}$ の空判定により判定できる．

3.3 段階的許容可能性

本節では，許容可能性と強充足可能性の関係に関する定理を示し，また，新たな実現可能性必要条件として**段階的許容可能性**を導入し，その判定手続きを与える．

許容可能性を提案した Damm ら [5] は指摘していないが，強充足可能性の定義から，支配的リアクティブシステムにおいて仕様を満たすようにふるまえなくてもよい入力列とは，明らかに悪性入力列である．

従って，以下の定理が成り立つ．

定理 19. LTL 式 φ について，φ が強充足可能かつ許容可能であるならば，かつそのときに限り，φ は実現

性質であるが,「段階的」と呼称される既存の実現可能性必要条件は何らかのリアクティブシステムの存在を示す性質として用いられているため，ここでは半強充足可能性と呼ぶ．

可能である．■

(**証明**) 定義より明らか．□

この定理から以下の系が導出される．

系 20. LTL 式 φ について，$\mathcal{L}(\varphi) \cup CST(\varphi)$ が実現可能であるならば，かつそのときに限り，φ は許容可能である．■

即ち，許容可能性は，悪性入力列（強充足可能性の反例）を許容するリアクティブシステムの存在を示す性質である．同様に，悪性入力接頭辞（半強充足可能性の反例）を許容するリアクティブシステムの存在を示す性質を考えることもできる．

そこで，この新しい実現可能性必要条件を段階的許容可能性と定義する．

定義 21 (段階的許容可能性)**.** LTL 式 φ について，$\mathcal{L}(\varphi) \cup CSEST(\varphi)$ が実現可能であるとき，φ は段階的許容可能であるという．

また，$\mathcal{L}(\varphi) \cup CSEST(\varphi)$ を実現するリアクティブシステムは φ に関して**段階支配的** (stepwise dominant) であるという．■

支配的リアクティブシステムはすべての無限長入力列に対して最適な出力列を応答するシステムであったが，段階支配的リアクティブシステムは各時刻において最適な出力を応答するシステムである．

段階的許容可能性の定義より，強充足可能性と許容可能性の関係に関する定理 19 と同様の定理が半強充足可能性と段階的許容可能性に関して成り立つ．

定理 22. LTL 式 φ について，φ が半強充足可能かつ段階的許容可能であるならば，かつそのときに限り，φ は実現可能である．■

(**証明**) 定義より明らか．□

従って，この段階的許容可能性判定手続きは以下のように与えられる．

手続き 23 (段階的許容可能性判定手続き)**.**

入力：LTL 式 φ

出力：φ が段階的許容可能か否か

1. 手続き 17（半強充足可能性判定手続き）のステップ 1–4 で得られる $\overline{\mathcal{A}_{\varphi}^{SeSt}}$ を決定化した上で，手続き 9（実現可能性判定手続き）のステップ 1 で構成した決定性オートマトンと和合成を行う．
2. その決定性和合成オートマトンに対して手続き 9 のステップ 2–3 を行い，実現可能ならば「段階的許容可能」と判定し，そうでないならば「段階的許容不能」と判定する．■

定理 24. 手続き 23 は正当な段階的許容可能性判定手続きである．■

(**証明**) 手続き 9 の正当性及び，定理 18 の証明中に示された $CSEST(\varphi)$ を受理するオートマトン $\overline{\mathcal{A}_{\varphi}^{SeSt}}$ の構成法の正当性より明らか．□

なお，許容可能性は，ステップ 1 で用いるオートマトン $\overline{\mathcal{A}_{\varphi}^{SeSt}}$ の代わりに，強充足可能性判定の際に用いられる $CST(\varphi)$ を受理するオートマトン $\overline{\mathcal{A}_{\varphi}^{St}}$ を利用することで判定できる．

4 階層定理

本章では，実現可能性必要条件の間の関係に関する定理を示す．なお，充足可能，半強充足可能，強充足可能，段階的充足可能，段階的強充足可能，真段階的充足可能，許容可能性，段階的許容可能性，実現可能な仕様の集合をそれぞれ Sat，$SeStSat$，$StSat$，$SwSat$，$SwStSat$，$PrSwSat$，Adm，$SwAdm$，$Real$ とする．

森らは，提案した実現可能性必要条件（強充足可能性，段階的充足可能性，段階的強充足可能性）の間の関係について，以下の定理を示している．

定理 25 ([15])**.** 以下の関係が成り立つ．

$$SwStSat \subset StSat \subset Sat, \tag{24}$$
$$SwStSat \subset SwSat \subset Sat, \tag{25}$$
$$StSat \setminus SwSat \neq \emptyset, \tag{26}$$
$$SwSat \setminus StSat \neq \emptyset, \tag{27}$$
$$Real \subset SwStSat \subset StSat \cap SwSat. \tag{28}$$

■

同様に，本稿で提案した真段階的充足可能性及び半強充足可能性と，実現可能性及び森らが提案した実現可能性必要条件との間の関係について，以下の定理が成り立つ．

定理 26. 以下の関係が成り立つ．

$$StSat \cup SwSat \subset SeStSat \subset Sat, \tag{29}$$
$$PrSwSat \subset SwSat, \tag{30}$$
$$SwSat \setminus (StSat \cup PrSwSat) \neq \emptyset, \tag{31}$$

$$PrSwSat \setminus StSat \neq \emptyset, \tag{32}$$
$$(StSat \cap SwSat) \setminus (SwStSat \cup PrSwSat) \neq \emptyset, \tag{33}$$
$$(StSat \cap PrSwSat) \setminus SwStSat \neq \emptyset, \tag{34}$$
$$SwStSat \setminus PrSwSat \neq \emptyset, \tag{35}$$
$$Real \subset SwStSat \cap PrSwSat. \tag{36}$$

■

(**証明**) 式 (29) と式 (30)，式 (36) の関係が部分集合であることは定義から明らかである．それらが真部分集合であること及び式 (31)–(35) は補題 27–34 より成り立つ．□

補題 27. 充足可能かつ半強充足不能な仕様が存在する．■

(**証明**) 以下の LTL 式 f_{27} を考える．

$$f_{27} = (req_1 \rightarrow res_1) \wedge (req_2 \rightarrow \neg res_1). \tag{37}$$

ただし，$req_1, req_2 \in \mathcal{I}$，$res_1 \in \mathcal{O}$ である．

このとき，初期入出力が $\{req_1, res_1\}$ であれば f_{27} は成り立つ（充足可能）．また，$\{req_1, req_2\}$ が f_{27} に対する悪性入力接頭辞である（強充足不能）．従って，f_{27} は充足可能かつ半強充足不能である．□

補題 28. 半強充足可能かつ強充足不能かつ段階的充足不能な仕様が存在する．■

(**証明**) 以下の LTL 式 f_{28} を考える．

$$f_{28} = \Box(res_1 \leftrightarrow \bigcirc req_1) \wedge \Diamond\Box(req_2 \rightarrow \neg res_1). \tag{38}$$

ただし，$req_1, req_2 \in \mathcal{I}$，$res_1 \in \mathcal{O}$ である．

このとき，接尾辞が $\{req_1\}^\omega$ である任意の入力列 $\tilde{a}$ に対して，f_{28} を成立させる（req_1 出現の 1 ステップ前に res_1 を置く）出力列が存在する（半強充足可能）．また，$\{req_1, req_2\}^\omega$ が f_{28} に対する悪性入力列である（強充足不能）．そして，左連言肢 $\Box(res_1 \leftrightarrow \bigcirc req_1)$ は典型的な段階的充足不能な式である．従って，f_{28} は半強充足可能かつ強充足不能かつ段階的充足不能である．□

補題 29. 段階的充足可能かつ強充足不能かつ真段階的充足不能な仕様が存在する．■

(**証明**) 以下の LTL 式 f_{29} を考える．

$$f_{29} = (req_1 \rightarrow \Diamond res_1) \wedge \Box(res_1 \rightarrow \Box req_2). \tag{39}$$

ただし，$req_1, req_2 \in \mathcal{I}$，$res_1 \in \mathcal{O}$ である．

このとき，$\emptyset$ を出力し続けるシステムは，初期入力が $\{req_1\}$ であっても充足可能性を保持し続けている（段階的充足可能）．これは，入出力接尾辞が $\{req_2, res_1\}^\omega$ であれば f_{29} が成り立つためである．また，$\{req_1\}^\omega$ が f_{29} に対する悪性入力列である（強充足不能）．そして，初期入力が $\{req_1\}$ であるとき，充足可能性を保持するためには $\neg res_1$ を出力し続けなければならない．これは，一度でも res_1 を出力してしまうと，それ以後 $\Box req_2$ を成立させなければならないが，システムは入力接尾辞を制御できないためである．しかし，充足可能性を保持するために $\neg res_1$ を出力し続けていれば $\Diamond res_1$ を成立させることはできない（真段階的充足不能）．従って，f_{29} は段階的充足可能かつ強充足不能かつ真段階的充足不能である．□

補題 30. 真段階的充足可能かつ強充足不能な仕様が存在する．■

(**証明**) 以下の LTL 式 f_{30} を考える．

$$f_{30} = (\Box\Diamond req_1 \rightarrow \Box\Diamond res_1) \wedge (\Diamond\Box req_2 \rightarrow \Diamond\Box\neg res_1). \tag{40}$$

ただし，$req_1, req_2 \in \mathcal{I}$，$res_1 \in \mathcal{O}$ である．

このとき，常に res_1 を出力し続けるシステムは，$\{req_1\}^\omega$ が接尾辞の任意の入力列に対して f_{30} を満たすようにふるまう（真段階的充足可能）．また，$\{req_1, req_2\}^\omega$ が f_{30} に対する悪性入力列である（強充足不能）．従って，f_{30} は真段階的充足可能かつ強充足不能である．□

補題 31. 強充足可能かつ段階的充足可能かつ段階的強充足不能かつ真段階的充足不能な仕様が存在する．■

(**証明**) 後述の式 (43) で与えられる LTL 式 f_{33} を部分式として含む，以下の LTL 式 f_{31} を考える．

$$f_{31} = (\Diamond\Box req_1 \leftrightarrow res_1) \wedge \bigcirc f_{33}. \tag{41}$$

ただし，$req_1 \in \mathcal{I}$，$res_1 \in \mathcal{O}$ である．

このとき，連言肢は意味的に互いに独立である．また，補題 33 の照明中で述べるように f_{33} は強充足可能かつ段階的充足可能かつ真段階的充足不能であり，右連言肢 $\bigcirc f_{33}$ と同様である．そして，左連言肢 $(\Diamond\Box req_1 \leftrightarrow res_1)$ に関しては，無限にしばしば req_1 が入力されるか否かに応じた初期出力が存在し（強充足可能），また，任意の初期出力に対しても f_{31} を成

立にする入力接尾辞が存在する（段階的充足可能）が f_{31} を不成立にする入力接尾辞も存在する（段階的強充足不能）．従って，f_{31} は強充足可能かつ段階的充足可能かつ段階的強充足不能かつ真段階的充足不能である． □

補題 32. 強充足可能かつ真段階的充足可能かつ段階的強充足不能な仕様が存在する． ■

(**証明**) 以下の LTL 式 f_{32} を考える．

$$f_{32} = \Box\Diamond req_1 \leftrightarrow res_1. \tag{42}$$

ただし，$req_1 \in \mathcal{I}$，$res_1 \in \mathcal{O}$ である．

このとき，無限にしばしば req_1 が出現する入力列に対しては $\{res_1\}^\omega$，そうでない入力列に対しては $\emptyset^\omega$ という出力列で f_{32} を満たす（強充足可能）．また，初期出力が $\emptyset$ であるシステムは，$\emptyset^\omega$ が接尾辞の任意の入力列に対して f_{32} を満たすようにふるまう（真段階的充足可能）．そして，初期出力が res_1 のとき $\emptyset^\omega$，そうでないとき $\{req_1\}^\omega$ が接尾辞の入力列に対して f_{32} を成立させる出力接尾辞が存在しない．(段階的強充足不能)．従って，f_{32} は強充足可能かつ真段階的充足可能かつ段階的強充足不能である． □

補題 33. 段階的強充足可能かつ真段階的充足不能な仕様が存在する． ■

(**証明**) 以下の LTL 式 f_{33} を考える．

$$f_{33} = (req_2 \rightarrow \Diamond res_1) \wedge \Box(res_1 \rightarrow \bigcirc\Box(\bigcirc req_2 \leftrightarrow res_1)). \tag{43}$$

ただし，$req_2 \in \mathcal{I}$，$res_1 \in \mathcal{O}$ である．

このとき，$\emptyset$ を出力し続けるシステムは強充足可能性を保持し続けている（段階的強充足可能）．これは，$\Box(\bigcirc req_2 \leftrightarrow res_1)$ が強充足可なためである．また，f_{29} と同様の理由で，f_{33} は真段階的充足不能である．従って，f_{33} は段階的強充足可能かつ真段階的充足不能である． □

補題 34. 段階的強充足可能かつ真段階的充足可能かつ実現不能な仕様が存在する． ■

(**証明**) 以下の LTL 式 f_{34} を考える．

$$f_{34} = \Box\Diamond req_1 \leftrightarrow \Diamond\Box res_1. \tag{44}$$

ただし，$req_1 \in \mathcal{I}$，$res_1 \in \mathcal{O}$ である．

このとき，req_1 が無限にしばしば出現する入力列に対しては $\{res_1\}^\omega$，そうでない入力列に対しては $\emptyset^\omega$ が接尾辞の出力列で，出力接頭辞に関わらず f_{34} を満たす（段階的強充足可能）．また，res_1 を出力し続けるシステムは，$\emptyset^\omega$ が接尾辞の入力列に対して，f_{34} を満たすようにふるまう真段階的充足可能）．しかしながら，res_1 を常に出力する/しないようになるタイミングを環境の協力なしに決定できない．(実現不能)．従って，f_{34} は段階的強充足可能かつ真段階的充足可能かつ実現不能である． □

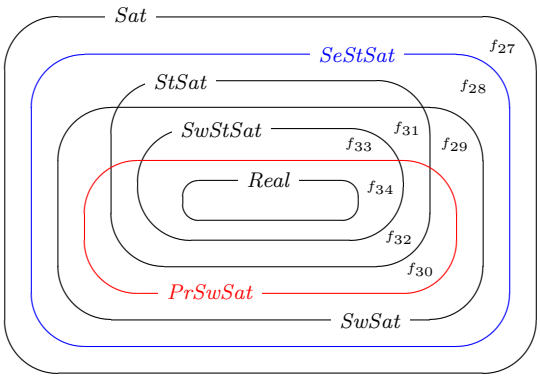

図 1　実現可能，真段階的充足可能，段階的強充足可能，段階的充足可能，強充足可能，半強充足可能，充足可能な仕様の集合（それぞれ $Real$，$PrSwSat$，$SwStSat$，$SwSat$，$StSat$，$SeStSat$，Sat）の包含関係．ただし，f_{27}–f_{34} は，それぞれ，補題 27–34 の証明中で挙げていた，その領域に含まれる式の例である．

定理 25 と定理 26 が示す包含関係をまとめると，図 1 のように与えられる．

また，充足可能性，半強充足可能性，強充足可能性，許容可能性，段階的許容可能性の間の関係に関しては，以下の定理が成り立つ．

定理 35. 以下の関係が成り立つ．

$$Sat^c \subset SwAdm \subset Adm, \tag{45}$$
$$StSat \setminus Adm \neq \emptyset, \tag{46}$$
$$SeStSat \setminus (StSat \cup Adm) \neq \emptyset, \tag{47}$$
$$Sat \setminus (SeStSat \cup Adm) \neq \emptyset, \tag{48}$$
$$(SeStSat \cap Adm) \setminus SwAdm \neq \emptyset, \tag{49}$$
$$(Sat \cap Adm) \setminus (SeStSat \cup SwAdm) \neq \emptyset, \tag{50}$$
$$(Sat \cap SwAdm) \setminus SeStSat \neq \emptyset. \tag{51}$$

ただし，^{c} は補集合演算である． ■

(**証明**) $SwAdm \subseteq Adm$ は定義より明らか．これが真部分集合であることは，式 (49) や式 (50) より成

り立つ．$Sat^c \subset SwAdm$ は，仕様 φ が充足不能ならば $CSEST(\varphi) = (2^{\mathcal{P}})^\omega$ であること及び実現可能ならば段階的許容可能であることから成り立つ．式 (46)–(51) は，以下の補題 36–41 より成り立つ． □

補題 36. 強充足可能かつ許容不能な仕様が存在する． ■

(**証明**) 強充足可能かつ段階的充足不能な以下の LTL 式 f_{36} を考える．

$$f_{36} = \Box(\bigcirc req_3 \leftrightarrow res_2). \tag{52}$$

ただし，$req_3 \in \mathcal{I}$，$res_2 \in \mathcal{O}$ である．

このとき，$CST(f_{36})$ が空集合であるため，f_{36} は許容不能である． □

補題 37. 半強充足可能かつ強充足不能かつ許容不能な仕様が存在する． ■

(**証明**) 半強充足可能かつ強充足不能な LTL 式 f_{28} と，強充足可能かつ段階的充足不能な LTL 式 f_{36} の連言 f_{37} を考える．

$$f_{37} = f_{28} \wedge f_{36}. \tag{53}$$

各連言肢は意味的に互いに独立であるため，それらの連言 f_{37} は明らかに許容不能である． □

補題 38. 充足可能かつ半強充足不能かつ許容不能な仕様が存在する． ■

(**証明**) 充足可能かつ半強充足不能な LTL 式 f_{27} と，強充足可能かつ段階的充足不能な LTL 式 f_{36} の連言 φ_{38} を考える．

$$f_{38} = f_{27} \wedge f_{36}. \tag{54}$$

各連言肢は意味的に互いに独立であるため，それらの連言 f_{38} は明らかに許容不能である． □

補題 39. 半強充足可能かつ許容可能かつ強充足不能かつ段階的許容不能な仕様が存在する． ■

(**証明**) 以下の LTL 式 f_{39} を考える．

$$f_{39} = \Diamond\Box req_1. \tag{55}$$

ただし，$req_1 \in \mathcal{I}$ である．

このとき，f_{39} は半強充足可能かつ強充足不能である．そして，$CSEST(f_{39})$ は空集合，$CST(f_{39})$ は $\mathcal{L}(f_{39})$ の補集合である．従って，f_{39} は半強充足可能かつ許容可能かつ強充足不能かつ段階的許容不能である． □

補題 40. 充足可能かつ許容可能かつ半強充足不能かつ段階的許容不能な仕様が存在する． ■

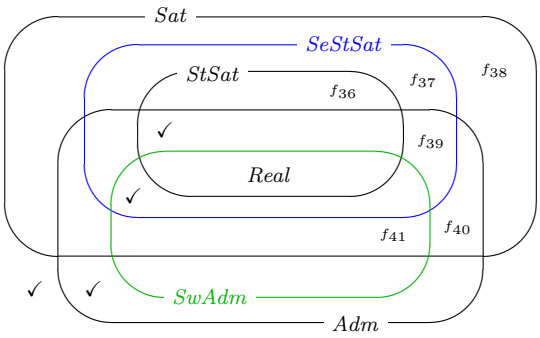

図 2　実現可能，強充足可能，段階的許容可能，半強充足可能，許容可能，充足可能な仕様の集合（それぞれ，$Real$，$StSat$，$SwAdm$，$SeStSat$，Adm，Sat）の包含関係．ただし，f_{36}–f_{41} は，それぞれ，補題 36–41 の証明中で挙げていた，その領域に含まれる式の例である．また，チェック記号 ✓ が付けられた領域は空集合である．

(**証明**) 半強充足可能かつ許容可能かつ強充足不能かつ段階的許容不能な LTL 式 f_{39} と，後述の式 (57) で与えられる充足可能かつ段階的許容可能かつ半強充足不能な LTL 式 f_{41} の連言 f_{40} を考える．

$$f_{40} = f_{39} \wedge f_{41}. \tag{56}$$

各連言肢は意味的に互いに独立であるため，この連言 f_{40} は充足可能かつ許容可能かつ半強充足不能かつ段階的許容不能である． □

補題 41. 充足可能かつ段階的許容可能かつ半強充足不能な仕様が存在する． ■

(**証明**) 以下の LTL 式 f_{41} を考える．

$$f_{41} = \Box req_2. \tag{57}$$

ただし，$req_2 \in \mathcal{I}$ である．

このとき，f_{41} は充足可能かつ半強充足不能である．そして，$CSEST(f_{41})$ は $\mathcal{L}(f_{41})$ の補集合である．従って，f_{41} は充足可能かつ段階的許容可能かつ半強充足不能である． □

定理 19 と定理 22，定理 35 が示す包含関係をまとめると，図 2 のように与えられる．

5　不完全合成

本章では，3. 1–3. 3 節で提案した実現可能性必要条件に基づいて，与えられた仕様から不完全なリアクティブシステムを合成する手法を与える．

5.1 真段階的充足可能性に基づいた不完全合成

本節では，真段階的充足可能な仕様から，式 (19) を満たすリアクティブシステムを合成する手続きを与える．

手続き 15（真段階的充足可能性判定手続き）のステップ 2 において，$\mathcal{A}_\varphi$ に対して 2 ステップの状態除去操作を変化がなくなるまで繰り返した結果の決定性オートマトンを $\mathcal{A}_\varphi^{PrSw} = \langle Q, 2^{\mathcal{P}}, \Delta, q_{init}, Acc\rangle$ とする．このオートマトン上の任意の状態は，受理実行に繰り返し到達可能，かつ，行き止まりでない．ここで，オートマトンの現在の状態が q であるとき入力 $a \subseteq \mathcal{I}$ に対して何を出力するかを決定する関数（即ち，戦略）$\mu : Q \times 2^{\mathcal{I}} \to 2^{\mathcal{O}}$ は，以下のリアクティブシステム r_μ と見なせる．

$$r_\mu(a_0 \cdots a_n) = \mu(q_n, a_n). \tag{58}$$

ただし，$q_0 = q_{init}$ かつ任意の $i < n$ について，

$$\{q_{i+1}\} = \Delta(q_i, a_i \cup \mu(q_i, a_i)). \tag{59}$$

$\mathcal{A}_\varphi^{PrSw}$ の構成法から，各状態及び各入力に対して遷移を可能とする出力が存在し，また，どの状態も初期状態から到達可能かつ受理強連結成分（受理 SCC）に到達可能である．よって，現在の状態が受理閉路上にある場合には受理閉路を辿り，そうでない場合には近づくように出力を決定する戦略 μ は，式 (19) を満たすリアクティブシステムと等価である．

そこで，各受理 SCC から受理閉路をちょうど 1 つずつ抽出する．ただし，それらの受理閉路上に現れる状態は重複しないものとし，抽出した各受理閉路上に現れる状態の集合を Q_{acc} とする．このとき，各状態から Q_{acc} へ到達するまでの距離 $D_{Q_{acc}}(q)$ は以下のように与えられる．

$$D_{Q_{acc}}(q) = \min\{n \mid q_n \in Q_{acc} \text{ and } q_0 q_1 \ldots q_n \in Path^{\mathcal{A}_\varphi^{PrSw}}(q)\}. \tag{60}$$

各状態 q において入力部分が $a \subseteq \mathcal{I}$ である場合に，$D_{Q_{acc}}(q')$ が最も小さい状態に遷移するための出力を選ぶ戦略，即ち，以下を満たす戦略 $\mu_{Q_{acc}}$ が目的の関数である．

$$\mu_{Q_{acc}}(q, a) = \operatorname*{argmin}_{b \subseteq \mathcal{O}} \{D_{Q_{acc}}(q') \mid q' \in \Delta(q, a \cup b)\}. \tag{61}$$

まとめると，真段階的充足可能な仕様からの不完全合成手続きは以下のように与えられる．

手続き 42 (真段階的充足可能性に基づいた不完全合成手続き)**.**

入力：真段階的充足可能な LTL 式 φ

出力：式 (19) を満たすリアクティブシステム

1. 手続き 15 のステップ 2 の状態除去操作を変化がなくなるまで繰り返し，その結果のオートマトン $\mathcal{A}_\varphi^{PrSw}$ を得る．
2. $\mathcal{A}_\varphi^{PrSw}$ の SCC 分解を行い，受理 SCC 毎に 1 つずつ受理閉路を抽出し，そこに現れる状態の集合 Q_{acc} 得る．
3. Q_{acc} へ到達するまでの距離 $D_{Q_{acc}}$ から，式 (61) を満たす戦略 $\mu_{Q_{acc}}$ （リアクティブシステム $r_{\mu_{Q_{acc}}}$）を生成し出力する． ■

定理 43. 手続き 15 と手続き 42 はそれぞれ正当な真段階的充足可能性判定手続きと式 (19) を満たすリアクティブシステムの合成手続きである． ■

(**証明**) $Q \neq \emptyset$ のとき，$\mathcal{A}_\varphi^{PrSw}$ 構成法から，Q に含まれる任意の状態は初期状態から到達可能かつ受理 SCC に到達可能であり，非空な Q_{acc} を必ず得ることができる．Q_{acc} が固定されていれば，$D_{Q_{acc}}$ と $\mu_{Q_{acc}}$ は一意に定まる．ここで，$\mathcal{A}_\varphi^{PrSw}$ が行き止まり状態をもたないことから，$r_{\mu_{Q_{acc}}}$ は全域的関数である．また，すべての状態から Q_{acc} へ到達可能もあるため，$D_{Q_{acc}}$ はすべての状態に対して有限の値をもっている．そして，Q_{acc} を構成する受理閉路が属する SCC は相異なるため，異なる受理閉路上の 2 つの状態が互いに到達可能となることはない．よって，任意の入力接頭辞に対して，$D_{Q_{acc}}$ を 0 まで減少させ（つまり，なんらかの受理閉路に到達し）その後 $D_{Q_{acc}}$ が 0 であることを保持する（つまり，その閉路を巡回し続ける）入力接尾辞が存在する．即ち，$r_{\mu_{Q_{acc}}}$ は式 (19) を満たすリアクティブシステムである．

$Q = \emptyset$ のとき，任意の入力列に対して，受理閉路へ繰り返し到達可能ないずれかの状態に留まらせる戦略が存在しない．よって，真段階的強充足不能である．

以上により示された． □

5.2 半強充足可能性に基づいた不完全合成

半強充足可能性は，なんらかのリアクティブシステムの存在を示す性質ではなく，また，充足可能性に次いで実現可能性から遠い．そこで，この性質と許容可能性を組み合わせて不完全合成に利用する．

系 20 から，仕様 φ の許容可能性判定は $\mathcal{L}(\varphi) \cup CST(\varphi)$ の実現可能性判定と同値である．この $\mathcal{L}(\varphi) \cup CST(\varphi)$ の実現可能性判定では，通常の LTL 実現可能性判定と同様に，$\mathcal{L}(\varphi) \cup CST(\varphi)$ を実現するリアクティブシステム（即ち，φ に対する支配的リアクティブシステム）を合成できる．

従って，仕様が許容可能であれば，その仕様に対する支配的リアクティブシステムを合成でき，さらにその仕様が半強充足可能ならば，そのシステムは悪性入力接頭辞をもたない悪性入力列のみを許容することを保証できる．即ち，そのシステムは，仕様に反するための選択を永久に行い続ける環境のみを許容する．

5.3 段階的許容可能性に基づいた不完全合成

定義 21 から，仕様 φ の段階的許容可能性判定は $\mathcal{L}(\varphi) \cup CSEST(\varphi)$ の実現可能性判定と同値である．この $\mathcal{L}(\varphi) \cup CSEST(\varphi)$ の実現可能性判定では，通常の LTL 実現可能性判定と同様に，$\mathcal{L}(\varphi) \cup CSEST(\varphi)$ を実現するリアクティブシステム（即ち，φ に対する段階支配的リアクティブシステム）を合成できる．

従って，仕様が段階的許容可能であれば，その仕様に対する段階支配的リアクティブシステムを合成できる．このシステムは，5.2 節で与えた半強充足可能性と許容可能性に基づいた不完全合成で得られるシステムとは異なり，悪性入力接頭辞をもつ入力列のみを許容するシステムである．即ち，そのシステムは，仕様に反することを有限時間以内に決定付けてしまう一連の選択を行う環境のみを許容する．

6 関連研究

6.1 実現可能性必要条件

最も基本的な実現可能性必要条件は充足可能性（無矛盾性）である．しかしながら，充足可能性と実現可能性の間の隔たりは非常に大きい．森らは，その隔たりを埋める，強充足可能性，段階的充足可能性，段階的強充足可能性を導入し [15] [27]，また，その判定手続きを与えている [23] [15] [27]．これらの性質及びその充足判定手続きにより，仕様が実現不能であった際に，その欠陥を分類することを可能とした．また，これら性質の判定問題は，実現可能性判定/自動合成（2EXPTIME 完全 [16] [18]）よりも計算量が小さく，仕様の修正・洗練を効率的に行うことも可能としている．判定の効率化も盛んに研究されており，安藤らは分散計算による効率的な段階的充足可能性判定手続きを実装した [25]．また，島川らは，SAT ベースの有界強充足可能性判定を提案し [21]，さらに，強充足可能性判定の計算量が EXPSPACE 完全であることを示した [22]．これらの実現可能性必要条件充足判定では欠陥の種類を分類できるが，具体的に仕様のどの箇所に欠陥があるかを特定できない．そのため，萩原らは，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た [10]．

一方，本研究では，新たに 3 つの実現可能性必要条件を提案し，その充足判定手続きを与えている（3 章各節）．また，既存のものも含め，実現可能性必要条件の間の関係についての定理を示した（4 章）．これにより，これまでよりも詳細な仕様欠陥解析が可能となる．提案した判定手続きは，既存実現可能性必要条件充足判定手続きと類似のものであるため，既存の効率化・応用技術を適用することも可能である．

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 9）では，ステップ 1 において，LTL2dstar[†4] [12] で実装されている Safra の構成法 [19] や，Rabinizer3[†6] で実装されている Esparza-Křetínský 法 [†5] [8] 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[†5] な手続き [13] [20] [9] がいくつも提案されており，Lily[†9] [11]，Unbeast[†10]

†9 http://www.iaik.tugraz.at/content/research/design_verification/lily/

†10 http://www.react.uni-saarland.de/tools/unbeast/

[7]，Acacia+[†11] [4]，[26] などの合成器が開発された．

本研究では，これらの手続き/ツールが目的とする，仕様を厳密に実現する完全なリアクティブシステムの合成ではなく，新たに導入した 3 つの実現可能性必要条件に基づいた不完全な合成を扱っている（5 章）．吉浦らは段階的充足可能性に基づいた不完全合成手法を提案していた [23]．その手法で得られるシステムは，3.1 節でも述べたように，どのような入力列に対しても仕様を満たすようにはふるまうことができない可能性がある．それに対して，5.1 節の真段階的充足可能性に基づいた不完全合成では，仕様を満たすような入力列が必ず保証された，より実現に近いシステムを合成できる．また，Damm らは，許容可能性に基づいた不完全合成手法を提案していた [5]．その不完全合成手法は悪性入力列を許容するシステムを合成するものであり，不完全合成された各部分システムを不備が露呈しないように組み合わせ，全体として安全な分散システムの構成する応用手法も提案されている [5]．一方，5.2 節と 5.3 節で与えた不完全合成では，それぞれ許容するふるまいを「悪性入力接頭辞をもたないもの」と「悪性入力接頭辞をもつもの」にさらに限定したシステムを合成できる．

7 おわりに

7.1 結論

本研究では，以下の 3 つの新しい実現可能性必要条件を導入した．

- 真段階的充足可能性
- 半強充足可能性
- 段階的許容可能性

また，これらの新たな必要条件に関して，既存の実現可能性必要条件充足判定手続きと類似の充足判定手法も与えた．さらに，既存のものも含め，実現可能性必要条件間の階層定理を証明した．本研究で導入した必要条件は既存の必要条件及び実現可能性の間の隔たりを埋めるものであり，より詳細な仕様欠陥分析が可能となる．

†11 `http://lit2.ulb.ac.be/acaciaplus/`

そして，実現可能性必要条件に基づいた一定の水準を満たすシステムを合成する不完全合成手法を提案した．これにより，仕様が実現不能であっても，仕様に反するふるまいをある程度許容するシステムを合成できる．

7.2 今後の課題

仕様に欠陥がなかったとしても，その規模が大きければ現実的な時間では実現可能性判定/合成することはできない．そのため，現実的な問題に対応するためには，仕様が実現可能か否かに関わらず，尤もらしくふるまうシステムを合成する手法の開発は非常に重要である．それが今後の課題の 1 つである．

また，現実的な問題を扱うためには，実現可能性判定の問題設定には改良の余地がある．実現可能性判定の問題設定では，「外部環境は仕様に反するよう敵対的にふるまう」という仮定を置いているに等しい．実際，実現可能性判定/合成では，システムと環境の相互作用は敵対的なゲームとしてモデル化されている．しかしながら，なんらかのサービスを提供するシステムとそのサービスを利用するユーザ（外部環境）の関係のように，必ずしもシステムと環境の目的が相反しない場合にはその相互作用は協力ゲーム（非零和ゲーム）としてモデル化することが妥当である．そのような相互作用する主体毎に異なる目的をもち得る場合の一般化実現可能性に関する研究も今後の課題である．

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] 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.

[3] Bloem, R., Ehlers, R., Jacobs, S., and Könighofer, R.: How to Handle Assumptions in Synthesis, *Proceedings 3rd Workshop on Synthesis*,

Chatterjee, K., Ehlers, R., and Jha, S.(eds.), Electronic Proceedings in Theoretical Computer Science, Vol. 157, Open Publishing Association, 2014, pp. 34–50.

[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] 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.

[6] 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.

[7] Ehlers, R.: Symbolic Bounded Synthesis, *Computer Aided Verification*, Touili, T., Cook, B., and Jackson, P.(eds.), Lecture Notes in Computer Science, Vol. 6174, Springer Berlin Heidelberg, 2010, pp. 365–379.

[8] 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.

[9] 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.

[10] 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.

[11] Jobstmann, B. and Bloem, R.: Optimizations for LTL Synthesis, *Formal Methods in Computer Aided Design, 2006. FMCAD '06*, nov. 2006, pp. 117–124.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

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

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] Yoshiura, N. and Yonezaki, N.: Program Synthesis for Stepwise Satisfiable Specification of Re-

active System, *Principles of Software Evolution, 2000. Proceedings. International Symposium on*, 2000, pp. 58–67.
[25] 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.
[26] 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.
[27] 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.