T2R2 東京科学大学 リサーチリポジトリ
Science Tokyo Research Repository

論文 / 著書情報
Article / Book Information

題目(和文)	リアクティブシステム仕様の実現可能性検証における計算量削減方式に関する研究
Title(English)	
著者(和文)	島川昌也
Author(English)	Masaya Shimakawa
出典(和文)	学位:博士（工学）, 学位授与機関:東京工業大学, 報告番号:甲第9664号, 授与年月日:2014年9月25日, 学位の種別:課程博士, 審査員:米崎　直樹,佐伯　元司,権藤　克彦,渡部　卓雄,西崎　真也
Citation(English)	Degree:, Conferring organization: Tokyo Institute of Technology, Report number:甲第9664号, Conferred date:2014/9/25, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Powered by T2R2 (Science Tokyo Research Repository)

リアクティブシステム仕様の
実現可能性検証における計算量削減方式
に関する研究

東京工業大学
大学院情報理工学研究科 計算工学専攻
島川 昌也

目 次

第 1 章　はじめに **3**
1.1　本研究の背景 3
1.2　研究の目的・内容 5
1.3　本論文の構成 8

第 2 章　リアクティブシステム仕様の実現可能性 **10**
2.1　リアクティブシステム 10
2.2　リアクティブシステムの動作仕様 11
2.3　実現可能性 12

第 3 章　強充足可能性判定問題の計算量 **14**
3.1　強充足可能性 14
3.2　準備：非決定性 Büchi オートマトン 15
3.3　計算量の上界 16
3.4　計算量の下界 17
3.5　時間演算子のネストの深さを制限した場合の計算量 22
3.6　議論 25

第 4 章　有界強充足可能性判定 **27**
4.1　有界強充足可能性 28
4.2　有界強充足可能性判定 30
4.3　SAT ソルバを用いた NBA の k-全受理判定 31
4.3.1　k-loop が不受理であることの特徴付け 31
4.3.2　SAT 問題への帰着 36
4.3.3　インクリメンタルな k-全受理判定法 38
4.3.4　他の行程グラフ構成を基にした帰着 42
4.4　有界強充足可能性判定問題の計算量 43
4.4.1　上界 44
4.4.2　下界 44

4.5 実験 49
4.5.1 実験内容 49
4.5.2 結果 50
4.6 議論 53

第 5 章 実現可能性判定に適した LTL サブセット 55
5.1 LTL サブセット $\mathrm{LTL}^{\mathrm{ep}}$ と $\mathrm{LTL}^{\mathrm{gp}}$ 56
5.2 ω オートマトン 59
5.3 $\mathrm{LTL}^{\mathrm{gp}}$ の非決定性 ω オートマトンの特徴 60
5.3.1 LTL から非決定性 ω オートマトンの構成 60
5.3.2 $\mathrm{LTL}^{\mathrm{gp}}$ から得られる非決定性 ω オートマトンの特徴 63
5.4 構造的特徴を持つ NBA（NCA）の決定化 66
5.4.1 前処理 66
5.4.2 決定化 68
5.5 決定性 ω オートマトンの構成 70
5.5.1 $\mathrm{LTL}^{\mathrm{gp}}$ からの決定性 ω オートマトンの構成 71
5.5.2 $\mathrm{LTL}^{\mathrm{ep}}$ からの決定性 ω オートマトンの構成 71
5.5.3 $\mathrm{LTL}^{\mathrm{ep}}$ と $\mathrm{LTL}^{\mathrm{gp}}$ の組み合わせ仕様からの決定性 ω オートマトンの構成 72
5.6 実現可能性判定 74
5.6.1 無限ゲーム 74
5.6.2 無限ゲームを用いた実現可能性判定 76
5.7 実験 76
5.7.1 実験内容 77
5.7.2 結果 77
5.8 議論 78

第 6 章 まとめ 87

謝辞 90

参考文献 91

付録 103

第1章　はじめに

1.1　本研究の背景

近年，情報システムはますます複雑化し，その制御プログラムは膨大な数の制御状態を持つ．そのため，あらゆる状況を想定した完全な安全性を保証することは単純なテストのみでは難しい．システムが設計時に予期していなかった状況に陥り，障害が発生することも珍しくない．車載制御システム，航空管制システムや原子力発電所の制御システムなどの欠陥は甚大な経済的・人的被害を生む．それゆえ，そのようなセイフティクリティカルなシステムについては，理論的な安全性が保証される形式検証技術への期待が高まり，高コストにも関わらず実際の開発でも活用されはじめている．

セイフティクリティカルなシステムの多くは，リアクティブシステムとしてみることができる．リアクティブシステムとは，環境とのインタラクションの維持を目的としたシステムのことであり，環境からのあらゆる要求に対して，適切なタイミングで正しく応答を返すことが強く求められるのが特徴である．

このようなシステムの検証において最も利用されている形式検証技術として，モデル検査 [23] がある．モデル検査とは，形式的に表現されたシステム（のモデル）に欠陥がないかを調べる技術であり，設計段階での検証や実装段階での検証において活用されている．モデル検査では，システムが別途記述された仕様を満たすかなどが検査される．システムの記述には，状態遷移系，プロセス代数（CSP[41], CCS[61] や π-計算 [67]），時間オートマトン [2]，確率的システム（マルコフ連鎖やマルコフ決定過程）などが用いられ，システムが満たすべき性質を表す仕様の記述には，線形時間論理（Linear Temporal Logic, LTL）[70]，分岐時間論理（Computational Tree Logic, CTL）[33] やそれらの拡張論理など, 時間論理が用いられることが多い．代表的なツールとしては，SPIN[42]，NuSMV[22]，Kronos[18]，PRISM[54] などがある．

モデル検査技術が設計・実装段階での検証に活用されている一方，より上流の仕様段階での検証に活用される技術についても研究が行われている．形式的な言語で厳密な仕様を記述し，分析することは，開発初期段階での欠陥の発見につながり，大きな手戻りを削減する．それゆえ，仕様段階での十分な検証は，設計・実装段階での検証以上に開発プロセス全体の効率向上に寄与するといわれている [43].

リアクティブシステム仕様の検証においては，時間論理で振る舞いの集合を定める動作仕様を記述し，検証することが有効である．安全なシステム構築のためにリアクティブシステム仕様に求められることは，単に充足可能である（仕様を満たす振る舞いが存在する）ことだけではない．環境に対して開いたリアクティブシステムにおいては，その振る舞いは環境の振る舞いに依存し，環境の振る舞いをシステムはコントロールできない．そのため，リアクティブシステム仕様は,「環境からのどのような要求イベントがどのような順序で生起しても，仕様を満たすような応答を過去の要求履歴より決定する実現システムが存在する」という実現可能性 [1, 71] を満たすことが望まれる．実現可能性の検査を行うことで，仕様記述において見過ごされがちな危険な状況に陥る可能性を検出することが可能となる．さらに，実現可能であることがわかれば，システムの合成も可能であり [71]，有限状態遷移機械で表現される実現システム（そのすべての振る舞いが仕様を満たすことが保証されている）を得ることができる．モデル検査と対比すると，

モデル検査：
入力：システム，仕様
出力：システムが仕様を満たすか？［Yes/No］
（システムが仕様の実現システムであるか？）

実現可能性判定・システム合成：
入力：仕様
出力：仕様が実現システムを持つか？［Yes/No］ ＋ 実現システム

のようになる．すなわち，モデル検査において検査対象となるシステムを，実現可能性判定・システム合成では機械的に合成することができる．

実現可能性やシステム合成について，これまでに様々な研究が行われている．仕様記述言語の拡張 [11, 26, 17, 92] や，非同期システムや分散システムを取り扱うための概念の拡張 [72, 73, 95, 57, 50] についての研

究がある．また，実現不能な仕様の分析を目的とした仕様の分類法 [64]，欠陥原因の特定手法 [65, 37] の研究も行われている．効率化に関しては，判定・合成手続きの単純化 [45, 52]，シンボリックな判定・合成法 [69, 29] や分割判定・合成法 [49, 35] などについての研究がある．ツールの開発も行われており，T3[9, 8, 5, 93]，Lily[45]，Acacia[34]（及び，その後継ツール Acacia+[16]）や Unbeast[30] などがある．

しかしながら，実現可能性判定やシステム合成は，一般に極めて煩雑で計算コストの高い処理を伴うという問題がある．LTL で記述されたリアクティブシステム仕様の実現可能性判定問題は 2EXPTIME 完全であることが知られている [74]．これまでに開発されているツールで検証できる仕様の規模は，ごく限られたものとなっている．このような問題が，実現可能性判定やシステム合成の実用化への大きな障壁となっている．

1.2 研究の目的・内容

本論文は，リアクティブシステム仕様の実現可能性に関する検証の計算量削減方式に関して，(i) 検証する性質の制限と (ii) 仕様の構文制限の２つのアプローチによる研究を行ったものである．

実現可能性判定問題は 2EXPTIME 完全であるが，それだけを理由に現実的でないと結論付けるべきではない．その計算量はあくまでワーストケースでの難しさを表しているに過ぎない．NP 完全の問題を扱う SAT ソルバは，近年驚くほど高速である．また，NONELEMENTARY（2EXPTIME よりも難しい問題のクラス）の問題を扱うモデル検査器 MONA[32] は，実際的な問題に適用されている．入力や調べる性質を制限するなどして問題を単純化すれば，多くの場合安価に判定できる可能性は十分にある．本研究は，このようなアプローチで実現可能性検証の適用範囲を広げることを目的としている．

(i) 検証する性質の制限

検証する性質を制限すれば，リアクティブシステム仕様の実現可能性に関する検証の困難さは避けられえる．適切に制限した検証性質（実現可能性の必要条件）であれば，その判定で実際的な仕様の欠陥の多くを検出できるため，そのような検証性質の導入，及びその効率的な判定手続きの構築は実用上重要である．

本研究では，実現不能な仕様の分類のために森らによって導入された強充足可能性 [64] という性質に着目する．強充足可能性とは,「任意の要求イベント列に対して，その仕様を満たす応答イベントが存在する」という性質である．これまでに，[97, 38] で判定手続きが，[65, 37] でこの性質が満たされないときの原因の導出法が提案されている．この性質は，実現可能性の必要条件ではあるが，実用的なリアクティブシステムの仕様の多くが，強充足可能ならば実現可能でもあるとの議論もあり [65]，この性質の判定によって，リアクティブシステム仕様の多く欠陥を検出できると考えられている．

まず本研究では，強充足可能性判定問題の計算量について考察し，

- LTL で記述されたリアクティブシステム仕様の強充足可能性判定問題は EXPSPACE 完全であること，及び
- 時間演算子のネストを 2 までと制限しても，強充足可能性判定問題の計算量は下がらず，EXPSPACE 完全のままであること

を示す．それが EXPSPACE 完全であることより，強充足可能性判定問題は，充足可能性判定問題（計算量は PSPACE 完全 [88]）よりも真に難しく，実現可能性判定問題（計算量は 2EXPTIME 完全）以下の難しさであることがわかる[1]．実現可能性と比較して計算量理論的な優位性があること，及び実用的なリアクティブシステムの仕様の多くが強充足可能ならば実現可能でもあることから，リアクティブシステム仕様の欠陥を効率的に検出するために，実現可能性判定の代わりに強充足可能性判定を用いることは有効であると考える．

さらに本研究では，強充足可能性を基にした有界検査手法を提案する．有界検査とは，入力としてパラメータ k も与え，その k に応じた反例（もしくは証拠）が存在するかを調べることである．このような検査手法は，有界モデル検査 [14] や Alloy Analyzer[44] など，他の検証の分野では広く受け入れられている．このような手法の長所は，単純な欠陥が存在するかを安価に検査できることにある．これは，

- 検査における探索範囲が限定さていること，及び
- 高速な SAT ソルバ（[66, 28, 13] など）や SMT ソルバ（[27] など）をその検査において効果的に利用できること

[1] P と NP との関係と同様，EXPSPACE $\neq$ 2EXPTIME であるかは明らかになっていない．

による．不具合のある実際的な検査対象には小さな反例が存在することが多く，制限された範囲の検査でも多くの欠陥を見つけられる．

ここでは，強充足可能性を制限した有界強充足可能性の概念を導入し，その判定手続きを提案する．有界強充足可能性は，強充足可能性をサイズ k の繰り返し構造として表現される要求イベント列のみを考慮するように制限した性質である．つまり，仕様，パラメータ k に対して,「サイズ k の繰り返し構造で表現される任意の要求イベント列に対して，仕様を満たす応答イベント列が存在する」という性質であり，仕様を満たすように応答できない単純な要求パターンが存在しないことを表す．強充足不能な仕様は，比較的に小さな反例（仕様を満たすように応答することができないサイズ k の繰り返し構造で表現される要求パターン）を持つことが多い．それゆえ，この性質の判定で多くの欠陥を検出できると予想される．ここでは，SAT ソルバを用いて有界強充足可能性を判定する手続きを与える．この手続では，まず，仕様を満たすように応答可能な要求イベント列をちょうど受理する非決定性 Büchi オートマトン (NBA) を構成する．そして，その NBA がサイズ k の任意の繰り返し構造で表現される要求イベント列を受理する（有界全受理である）かどうかを調べる．本研究では，この NBA の有界全受理判定を SAT ソルバを用いて行う．

有界強充足可能性問題の計算量についても考察し，LTL で記述されたリアクティブシステム仕様の有界強充足可能性判定問題が co-NEXPTIME 完全であることを示す．これにより，有界強充足可能性判定問題が強充足可能性判定問題以下の難しさであることを明らかにする[2]．

本研究では，実験により，本手法が強充足可能性や実現可能性の判定と比べてより大きな仕様を取り扱え，小さな繰り返し構造として表現される反例の存在を効率的に調べられることを確かめた．本論文では，その結果についても報告する．

(ii) 仕様の構文制限

検証する仕様の構文を制限すれば，検証の煩雑さは避けられえる．少ないコストで判定を行え，十分な表現力を持つ言語のサブセットを見つけ出すことは実用上の観点から重要である．本研究では，実現可能性判定の煩雑さを避けられるように構文を制限した LTL のサブセットを与える．

[2] co-NEXPTIME $\neq$ EXPSPACE であるかは明らかになっていない．

LTL で記述されたリアクティブシステム仕様の実現可能性判定は，通常，仕様から決定性 ω オートマトンを構成し，それを分析することで行われる．決定性 ω オートマトンは，一般に Safra の構成法 [75] を用いて非決定性 ω オートマトンを決定化することで構成される．この Safra の構成法は，有限長の語の上のオートマトンの決定化で用いられる部分集合構成法に比べ，極めて煩雑で計算コストの高い処理を伴う．部分集合構成法では，元のオートマトンの状態集合を決定化後の状態とするのに対し，Safra の構成法では，元のオートマトンの状態集合をノードとする木を決定化後の状態とする．このような煩雑な構成のため，各種効率化を施しづらい．過去に実装も試みられているが，状態が十数個のオートマトンの決定化が限界である [40]．これまでに Safra の構成法を用いない実現可能性判定法も提案されいているが [52]，取り扱うことのできる仕様の規模は依然としてごく限らたものとなっている．

そこで本研究では，決定性 ω オートマトン構成を単純化する観点から構文を制限した LTL のサブセット $\mathrm{LTL}^{\mathrm{ep}}$ とその双対である $\mathrm{LTL}^{\mathrm{gp}}$ を，決定性 ω オートマトン構成法とともに提案する．$\mathrm{LTL}^{\mathrm{gp}}$ は，その仕様からある構造的な特徴を持つ非決定性 ω オートマトンを構成できるという性質を持つ．提案する構成法では，その構造的特徴を利用し，Safra の構成法よりも簡潔な部分集合構成法で非決定性 ω オートマトンを決定化することが可能である．$\mathrm{LTL}^{\mathrm{ep}}$ についても，$\mathrm{LTL}^{\mathrm{gp}}$ との双対性を利用することで，部分集合構成法を基に決定性 ω オートマトンを構成できる．

本研究では，構文を制限した仕様について実験を行い，本手法によってより大規模な仕様を取り扱え，効率的に実現可能性判定を行えることを確かめた．本論文では，その結果についても報告する．

1.3 本論文の構成

本論文の構成は次の通りである．まず，2 章で，リアクティブシステム仕様の実現可能性に関する諸概念の定義を与える．次に，3 章で[3]，LTL で記述されたリアクティブシステム仕様の強充足可能性判定問題の計算量が EXPSPACE 完全であること，及び時間演算子のネストを 2 までと制限してもその計算量は変わらないことを示す．次に，4 章で[4]，有界強充足可能性の概念を導入し，SAT ソルバを用いた判定手続きを与える．その有界強

[3] 3 章は，[80, 83, 84] を基にしている．
[4] 4 章は，[81, 85, 86] を基にしている．

充足可能性判定問題の計算量についても考察し，それが co-NEXPTIME 完全であることを示す．さらに，提案する手法と他の性質判定の実行時間を比較する実験の結果を示す．次に，5 章で[5]，実現可能性判定で必要となる決定性 ω オートマトン構成を簡単化する観点から構文を制限した LTL のサブセットを与える．そのサブセットの特徴について述べ，その特徴を利用する部分集合構成法を基にした決定性 ω オートマトン構成法を提案する．さらに，提案する手法と他の手法の効率を比較する実験の結果を示す．最後に 6 章で，本論文のまとめを述べる．

[5] 5 章は，[79, 82] を基にしている．

第2章　リアクティブシステム仕様の実現可能性

本章では，リアクティブシステム仕様の実現可能性に関する諸概念の定義を与える．ここで与える定義は，基本的に [64] に準じている．

2.1　リアクティブシステム

リアクティブシステムとは，環境とのインタラクションの維持を目的とするシステムであり，$RS = \langle X, Y, r \rangle$ と形式化できる．ここで，

- X は環境が生起させる要求イベントの集合である．
- Y はシステムが生起させる応答イベントの集合である．
- $r : (2^X)^+ \to 2^Y$ は，リアクション関数である．ここで，$(2^X)^+$ は有限長の要求イベント集合の列の集合である（i 番目の要素は，時点 i で生起したイベントの集合であり，そこに含まれないイベントはそこで生起しなかったことを表す）．リアクション関数とは，以前に生起した要求イベントの集合の列から現時点においてシステムが生起させる応答イベントの集合を決定する関数である．

要求イベント集合 X の部分集合の列を要求イベント集合列と呼び，応答イベント集合 Y の部分集合の列を応答イベント集合列と呼ぶ．また，イベント集合 $X \cup Y$ の部分集合の無限長の列を振る舞いと呼ぶ．

要求イベント集合列 $\tilde{a} = a_0 a_1 a_2 \ldots$ に対するリアクティブシステム $RS = \langle X, Y, r \rangle$ の振る舞いは，

$$(a_0 \cup r(a_0))(a_1 \cup r(a_0 a_1))(a_2 \cup r(a_0 a_1 a_2)) \ldots$$

であり，$behave_{RS}(\tilde{a})$ と記す．

2.2 リアクティブシステムの動作仕様

リアクティブシステムの動作仕様は，システムの可能な無限長の振る舞いの集合を定める．可能な振る舞いを定める手段としては様々なものがあるが，本論文では，リアクティブシステムの仕様記述言語として線形時間論理（Linear Temporal Logic, 以下 LTL）を用いる．

LTL の構文

本論文で扱う LTL は，時間演算子として neXt 演算子 $\mathbf{X}$ と強い Until 演算子 $\mathbf{U}$ を持つものである．

要求イベントの集合 X と応答イベントの集合 Y が与えらたとき，LTL 式は以下のように定義される．

- $p \in X \cup Y$ ならば，p は LTL 式である．
- φ, ψ が LTL 式ならば，$\neg\varphi$, $\mathbf{X}\varphi$, $\varphi \wedge \psi$, $\varphi\mathbf{U}\psi$ も LTL 式である．

LTL の意味論

LTL 式は，振る舞い, つまり $X \cup Y$ の部分集合の無限列上に解釈される．φ を LTL 式とし，$\sigma \in (2^{X \cup Y})^\omega$ を振る舞いとする．振る舞い σ が φ を満たすことを $\sigma \models \varphi$ と表し，以下のように再帰的に定義する．

- $\sigma \models p$ **iff** $p \in \sigma[0]$,
- $\sigma \models \neg\varphi$ **iff** $\sigma \not\models \varphi$,
- $\sigma \models \varphi \wedge \psi$ **iff** $\sigma \models \varphi$ and $\sigma \models \psi$,
- $\sigma \models \mathbf{X}\varphi$ **iff** $\sigma[1\ldots] \models \varphi$,
- $\sigma \models \varphi\mathbf{U}\psi$ **iff** $\exists j \geq 0\big(\sigma[j\ldots] \models \psi$ and $\forall k(0 \leq k < j)\, \sigma[k\ldots] \models \varphi\big)$.

ここで，$\sigma[i]$ は σ の i 番目の要素を，$\sigma[i\ldots]$ は σ の i 番目以降の振る舞いを表す．

LTL 式 φ に対して $\sigma \models \varphi$ を満たす σ が存在する場合，φ は充足可能であるという．

略記として，$\top, \bot, \vee, \rightarrow, \oplus$ を通常のように用いる．また，Release 演算子 $\mathbf{R}$（Until 演算子の双対），Finally 演算子 $\mathbf{F}$，及び Globally 演算子 $\mathbf{G}$（Finally 演算子の双対）を以下のように導入する．

$$\varphi \mathbf{R} \psi \equiv \neg(\neg\varphi \mathbf{U} \neg\psi)$$
$$\mathbf{F}\varphi \equiv \top \mathbf{U} \varphi$$
$$\mathbf{G}\varphi \equiv \neg\mathbf{F}\neg\varphi$$

各演算子の直感的な意味は，以下の通りである．

$\mathbf{X}\varphi$:「次の時間で φ が成立する」

$\varphi \mathbf{U} \psi$:「ψ がいつか必ず成立し，それまでの間ずっと φ が成立し続ける」

$\varphi \mathbf{R} \psi$:「ψ がずっと成立し続ける，もしくは ψ が成立しなくなる時点までに（少なくとも 1 度は）φ が成立する」

$\mathbf{F}\varphi$:「いつか必ず φ が成立する」

$\mathbf{G}\varphi$:「ずっと φ が成立し続ける」

2.3 実現可能性

実現可能性とは,「どのような要求イベントがどのような順序で生起しても，仕様を満たすような応答を過去の要求履歴から決定できるリアクティブシステムが存在する」という性質であり，次のように形式化される．

Spec を仕様とする．リアクティブシステム *RS* が,

$$\forall \tilde{a}.(behave_{RS}(\tilde{a}) \models Spec)$$

を満たすとき，*RS* は *Spec* の実現であるという．*Spec* が実現を持つとき，*Spec* は実現可能 [1, 71] であるという．

例

例として，以下のようなドア制御システムの仕様について考える．

0. ドアには，開ボタンと閉ボタンとがある．

1. 開ボタンが押されたら，いつか必ずドアは開く．

2. 閉ボタンが押されている間は，ドアは閉じ続ける．

「開ボタンが押される」と「閉ボタンが押される」を要求イベントとし，それぞれを x_1 と x_2 とで表す.「ドアを開く（閉じる)」を応答イベントとし，$y(\neg y)$ で表す．このとき，この仕様は次のように表現できる．

$$door : \mathbf{G}((x_1 \to \mathbf{F}y) \wedge (x_2 \to \neg y))$$

この仕様 $door$ は一見問題のないようにみえるが，実現可能ではない（充足可能ではある)．これは，開ボタンが押された以後ずっと閉ボタンが押され続ける場合，いかなるシステムも仕様を満たすようには振る舞えないためである．すなわち，$\tilde{a} = \{x_1, x_2\}\{x_2\}\{x_2\}\ldots$ に対して $behave_{RS}(\tilde{a}) \models door$ を満たす RS は存在しない．したがって，$\exists RS \forall \tilde{a}.(behave_{RS}(\tilde{a}) \models door)$ は満たされず，仕様 $door$ は実現不能となる．

条件 1 を次のように変更することを考える．

1' 閉ボタンが押されていないときに開ボタンが押されたら，ドアを開く．

仕様は次のように表現できる．

$$door' : \mathbf{G}((x_1 \wedge \neg x_2 \to y) \wedge (x_2 \to \neg y))$$

このとき，次のようなリアクション関数を持つ RS は，仕様 $door'$ の実現となりえ，

$$r(a_1 a_2 \ldots a_n) = \begin{cases} \{\} & x_2 \in a_n \\ \{y\} & \text{その他のとき} \end{cases}$$

仕様 $door'$ は実現可能である．

第3章　強充足可能性判定問題の計算量

本研究では，仕様に潜む欠陥を効率的に検出する手法を模索するにあたり，強充足可能性 [64] という性質に着目する．強充足可能性は，実現不能な仕様の分類のために森らによって導入された実現可能性の必要条件の一つである．この性質は，実現可能性の必要条件ではあるが，実際的なリアクティブシステム仕様の多くが，実現不能であれば強充足不能でもあるため [65]，この性質の判定によって，仕様の多くの欠陥を検出できると考えられている．

本章では，LTL で記述されたリアクティブシステム仕様の強充足可能性判定問題の計算量について考察し，それが EXPSPACE 完全であることを示す．すなわち，強充足可能性判定問題が決定性チューリング機械で $O(2^{p(n)})$ の領域（$p(n)$ は n の多項式関数）を使って解ける問題のクラス EXPSPACE に属すること，及び EXPSPACE に属する任意の問題が強充足可能性判定問題に多項式帰着可能であることを示す．さらに，強充足可能性判定問題は，時間演算子のネストを 2 までと制限しても，その計算量は EXPSPACE 完全のままであることも示す．

3.1　強充足可能性

本節では，実現可能性の必要条件である強充足可能性について紹介する．

強充足可能性とは，「環境が将来生起する要求イベント集合列が予めわかるならば，仕様を満たすように生起させる応答イベントの列を決定できる」という性質であり，次のように定義される．

定義 3.1.1 (強充足可能性 [64])**.** 仕様 *Spec* が以下を満たすとき，*Spec* は強充足可能であるという．

$$\forall \tilde{a} \exists b. (\langle \tilde{a}, \tilde{b} \rangle \models Spec)$$

ここで，$\tilde{a}$ は無限長の要求イベント集合列であり，$\tilde{b}$ は無限長の応答イベント集合列である．$\tilde{a} = a_0 a_1 \ldots$，$\tilde{b} = b_0 b_1 \ldots$ のとき，

$$\langle \tilde{a}, \tilde{b} \rangle = (a_0 \cup b_0)(a_1 \cup b_1) \ldots$$

とする．

例として，2.3 節で扱ったドア制御システムの仕様

$$door : \mathbf{G}((x_1 \rightarrow \mathbf{F}y) \wedge (x_2 \rightarrow \neg y))$$

について考える．仕様 $door$ は実現不能であるが，強充足不能でもある．これは，要求イベント集合列 $\tilde{a} = \{x_1, x_2\}\{x_2\}\{x_2\} \ldots$（開ボタンが押された以後ずっと閉ボタンが押され続けることを表す）に対して，$door$ を満たす応答イベント集合列が存在しない，すなわち，$\exists \tilde{b} \langle \tilde{a}, \tilde{b} \rangle \models door$ が満みたされないためである．

続いて，強充足可能性と実現可能性の違いをみるため，次のような仕様について考える．

$$\mathbf{F}r \leftrightarrow s$$

ここで，r は要求イベントであり，s は応答イベントである．r の如何なる生起パターンに対しても仕様を満たす応答パターンは存在するため，この仕様は強充足可能である．しかし，実現可能ではない．リアクティブシステムが観測できるのは，すでに生起したイベントのみである．この仕様のように将来の要求イベントの生起が分からなければ現在の出力を決定できない場合，仕様は実現不能となる．強充足可能性と実現可能性との違いは,「システムは環境の将来起こす要求を知ることができない」という制約を考慮しているかどうかにある．

しかし，$\mathbf{F}r \leftrightarrow s$ のような仕様が記述されることは稀であり，実用的なリアクティブシステムの仕様の多くが，強充足可能ならば実現可能でもある [65]．そのため，この強充足可能性の判定によって，リアクティブシステム仕様の多く欠陥を検出できると考えられている．

3.2 準備：非決定性 Büchi オートマトン

本研究では，強充足可能性判定問題の計算量を分析するにあたり，無限長の語を扱う ω オートマトンの一種である非決定性 Büchi オートマト

ン（Non-deterministic Büchi Automata，以下 NBA）を用いる[1]．以下では，その定義を与える．

NBA は，五つ組 $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ である．ここで，

- Σ は有限のアルファベット，
- Q は有限の状態の集合，
- $q_I \in Q$ は初期状態，
- $\delta \subseteq Q \times \Sigma \times Q$ は遷移関係，
- $F \subseteq Q$：受理状態の集合である．

Σ 上の無限長の語 α の行程は，以下を満たす状態の無限列 $\varrho \in Q^\omega$ である．

- $\varrho[0] = q_I$ であり，かつ
- すべての $i \geq 0$ において $(\varrho[i], \alpha[i], \varrho[i+1]) \in \delta$ である

行程が F に含まれる状態を無限回通過するとき，すなわち，$inf(\varrho) \cap F \neq \emptyset$ を満たすとき，その行程を受理行程と呼ぶ．ここで，$inf(\varrho)$ は Q 上の無限列 ϱ に無限回出現する要素の集合を表す．

無限長の語 α に対して，受理行程が存在するとき，$\mathcal{A}$ は α を受理するという．$\mathcal{A}$ が受理する無限長の語の集合を $\mathcal{A}$ の受理言語といい，$L(\mathcal{A})$ で表す．

3.3 計算量の上界

まず，強充足可能性判定問題の計算量の上界，すなわち，この問題が EXPSPACE に属することを示す．ここでは，$O(2^{p(n)})$ の領域を使って判定可能な強充足可能性手続きを与えることで，それを明らかにする．ここで与える手続きは，[38] によって提案された ω オートマトンを用いる強充足可能性判定手続きに変更を加えたものである[2]．

Spec を LTL によるリアクティブシステム仕様とする．*Spec* の強充足可能性の判定は，次の手順で行うことができる．

[1] 5 章では他の ω オートマトンも用いる．その定義は 5.2 節で与える．

[2] [38] では，Muller オートマトンを用いた判定手続きが提案されている．本論文では，非決定性 Büchi オートマトンによる判定手続きを定義する．

1. *Spec* を満たす振る舞いをちょうど受理する，つまり，$L(\mathcal{A}) = \{\sigma \mid \sigma \models \mathit{Spec}\}$ となる NBA $\mathcal{A} = \langle 2^{X \cup Y}, Q, q_I, \delta, F \rangle$ を構成する．

2. $\mathcal{A}$ をもとに，アルファベットを要求イベントのみに制限した NBA $\mathcal{A}' = \langle 2^X, Q, q_I, \delta', F \rangle$ を構成する．ここで，$\delta' = \{(q, a, q') \mid \exists b.(q, a \cup b, q') \in \delta\}$ である．

   - $L(\mathcal{A}') = \{\tilde{a} \mid \exists \tilde{b}.\langle \tilde{a}, \tilde{b} \rangle \models \mathit{Spec}\}$ となる．すなわち，$\mathcal{A}'$ は仕様を満たすように応答可能な要求イベント集合列をちょうど受理する．

3. $\mathcal{A}'$ の全受理判定（$L(\mathcal{A}') = (2^X)^\omega$ であるかの判定）を行う．全受理であるならば「強充足可能」と出力し，そうでない場合,「強充足可能でない」と出力する．

$\mathcal{A}$ は，[90] 等で与えられている通り，$\mathcal{O}(2^{|\mathit{Spec}|})$ の領域を用いて，$\mathcal{O}(2^{|\mathit{Spec}|})$ のサイズで構成可能である．$\mathcal{A}'$ は，$\mathcal{A}$ の遷移関係に射影操作を施すことで得られるため，$\mathcal{O}(|\mathcal{A}|)$ の領域を用いて，$\mathcal{O}(|\mathcal{A}|)$ のサイズで構成可能である．NBA の全受理判定は PSPACE である [89] ため，決定性チューリング機械で $\mathcal{O}(p(|\mathcal{A}'|))$ の領域を使って，ステップ 3 の判定を行える．したがって，決定性チューリング機械で $\mathcal{O}(2^{p(|\mathit{Spec}|)})$ の領域を使って，強充足可能性の判定を行える．よって，強充足可能性問題は EXPSPACE に属する．

定理 3.3.1. *LTL* によるリアクティブシステム仕様の強充足可能性問題は，*EXPSPACE* に属する．

3.4 計算量の下界

次に，強充足可能性判定問題の計算量の下界，すなわち，この問題が EXPSPACE 困難であることを示す．ここでは，その計算量が EXPSPACE 完全であると知られている EXP-CORRIDOR TILING 問題（[21, 15] など）が強充足可能性判定問題の補問題へ多項式帰着可能であることを示し，その EXPSPACE 困難性を明らかにする．

EXP-CORRIDOR TILING 問題とは，$\langle T, H, V, t_{init}, t_{final}, m \rangle$（ここで，$T$ は有限のタイル集合，$H, V \subseteq T \times T$ は横の隣接制約，縦の隣接制約であり，$t_{init}, t_{final} \in T$ は初期タイル，最終タイルである．m は 1 進数で表

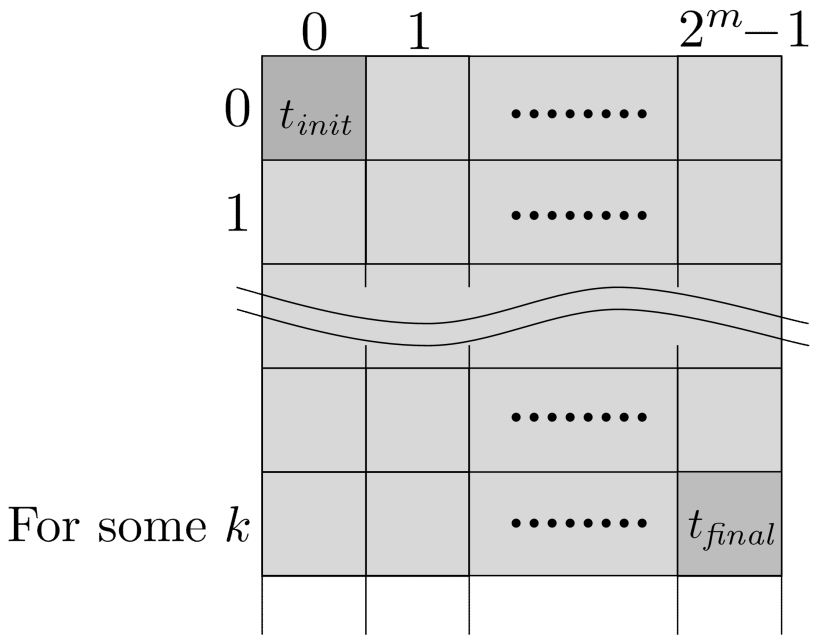

図 3.1: EXP-CORRIDOR TILING 問題

現された自然数である）に対して，ある $k \in \mathbb{N}$ が存在し，以下の条件を満たすタイルの割り当て関数 $f : [0, \ldots, (2^m - 1)] \times [0, \ldots, k] \to T$ が存在するかを決定する問題である．

- （初期タイルの条件）：$f(0, 0) = t_{init}$ である．
- （最終タイルの条件）：$f(2^m - 1, k) = t_{final}$ である．
- （横に隣接するタイルの条件）：任意の $0 \leq i < 2^m - 1$, $0 \leq j \leq k$ において，$(f(i, j), f(i + 1, j)) \in H$ である．
- （縦に隣接するタイルの条件）：任意の $0 \leq i \leq 2^m - 1$, $0 \leq j < k$ において，$(f(i, j), f(i, j + 1)) \in V$ である．

直感的には，図 3.1 に示すように，横が 2^m，縦が非有界の格子に対して，ある $k \in \mathbb{N}$ が存在し，$0 \leq i < 2^m$, $0 \leq j \leq k$ の各マス目 (i, j) に 4 つの条件，つまり，初期タイル条件（最左上のマス目は t_{init} である），最終タイル条件（最右下のマス目は t_{final} である），横の隣接条件（横に隣接するマス目に置かれるタイルは H を満たす），縦の隣接条件（縦に隣接するマス目に置かれるタイルは V を満たす）を満たすようにタイルを置くことができるかを判定する問題である．

上述のタイリング問題から強充足可能性判定問題の補問題への多項式帰着法，すなわち，タイリング問題 $\langle T, H, V, t_{init}, t_{final}, m\rangle$ に対して，その答えが「yes」であるとき，かつそのときにのみ，以下を満たすような φ_{tiling} の構成方法について述べる．

$$\exists\tilde{a}\forall\tilde{b}.(\langle\tilde{a},\tilde{b}\rangle \models \neg\varphi_{tiling})$$

ここで述べる帰着では，タイリング問題における「あるタイリングが存在する」を上式の「ある要求イベント集合列が存在する」に対応させる．そして,「そのタイリングが条件を満たす」を「そのタイリングに対応する要求イベント集合列については，任意の応答イベント集合列に対して LTL 式を満たすことがない」に対応させる．つまり，タイリング問題に対して，要求イベント集合列でその 1 つのタイリング（割り当て関数）を表現するように要求イベントを定める．そして，そのタイリングが条件を満たすことを，応答イベント集合列について全称束縛される式として表現するように応答イベント及び LTL 式を定める．

縦の隣接条件以外の表現においては，メタレベルの全称限量は必要ない．つまり，縦の隣接条件以外の条件は，要求イベントのみからなる（純粋な）LTL 式で表現される．これらの表現については，[24] で与えられている帰着での表現に近い．縦の隣接条件の表現においては，メタレベルの全称限量が必要となり，この表現が本帰着で最も重要な部分であるといえる．

要求イベント

要求イベントとしては，以下を用意する．

- x_t （$t \in T$）: 格子のマス目 (i, j) においてタイル t が置かれるかを，時点 $i + (2^m) \cdot j$ において x_t が生起しているかに対応させる．
- end : その時点でタイリングが終了しているかを表す．つまり，格子のマス目 (i, j) においてタイリングが終了しているかを時点 $i+(2^m)\cdot j$ において end が生起しているかに対応させる．
- $c_0, \ldots c_{m-1}$: m ビットのカウンタとして用いる．はじめは値が 0 で，1 単位時間ごとにその値が 1 づつ増えるようにし，その時点において着目しているマス目が何番目の列であるかを特定するのに用いる．

応答イベント

応答イベントとしては，以下を用意する．

- $y_0, \ldots y_{m-1}$：縦の隣接制約を記述する際に，着目する列を特定するのに用いる．

式： φ_{tiling}

式 φ_{tiling} を以下の式 (3.1)–(3.6) の論理積の否定とする．ただし，次を略記として用いる．

$$
\begin{aligned}
\overline{c} = 0 &\equiv \bigwedge_{0 \le i < m} \neg c_i \\
\overline{c} = 2^m - 1 &\equiv \bigwedge_{0 \le i < m} c_i \\
\overline{c} = \overline{y} &\equiv \bigwedge_{0 \le i < m} (c_i \leftrightarrow y_i)
\end{aligned}
$$

- m ビットカウンター $c_0, \ldots c_{m-1}$ の制約．

$$
\left(\bigwedge_{0 \le i < m} \neg c_i \right) \quad \wedge \quad \left(\bigwedge_{0 \le i < m} \mathbf{G} \left((c_i \oplus \bigwedge_{0 \le j < i} c_j) \ \leftrightarrow \ \mathbf{X} c_i \right) \right) \tag{3.1}
$$

「はじめは $\overline{c}$ の指す値が 0 であり，1 単位時間ごとにその値が 1 づつ増える」ことを表す．

- マス目に張られるタイルを一つにする制約．

$$
\bigwedge_{t \in T} \mathbf{G} \left(x_t \ \rightarrow \ \bigwedge_{t' \neq t} \neg x_{t'} \right) \quad \wedge \quad \mathbf{G} \left(\neg end \ \rightarrow \ \bigvee_{t \in T} x_t \right) \tag{3.2}
$$

「各マス目に張られるタイルは高々一つ，かつ，そこでタイリングが終了していないならば，いずれかのタイルが張られる」ことを表す．

- 初期タイルの制約．

$$
x_{t_{init}} \tag{3.3}
$$

「マス目 $(0, 0)$ にタイル t_{init} が置かれる」ことを表す．

- 最終タイルの制約.

$$\neg end \mathbf{U}\Big(\neg end \wedge \overline{c} = 2^m - 1 \wedge x_{t_{final}} \wedge \mathbf{XG}\, end\Big) \tag{3.4}$$

「$2^m - 1$ の列のどこかのマス目に最終タイルが置かれ，タイリングが終了する」こと表す.

- 横の隣接制約.

$$\mathbf{G}\Big(\overline{c} \neq 2^m - 1 \wedge \neg end \rightarrow \bigvee_{(t,t') \in H} \big(x_t \wedge \mathbf{X} x_{t'}\big)\Big) \tag{3.5}$$

「タイリングが終了しておらず，かつ，現マス目が $2^m - 1$ の列でない（右にマス目が存在する）ならば，現マス目に置くタイルと右マス目に置くタイルとが横の隣接制約を満たす」ことを表す.

- 縦の隣接制約.

$$\begin{aligned}
&\Big(\bigwedge_{0 \leq i < m} \mathbf{G}(y_i \leftrightarrow \mathbf{X} y_i)\Big) \rightarrow \\
&\mathbf{G}\Big(\Big(\overline{c} = \overline{y} \wedge \mathbf{XF}(\neg end \wedge \overline{c} = 0)\Big) \rightarrow \\
&\bigvee_{(t,t') \in V} \Big(x_t \wedge \mathbf{X}\big((\overline{c} \neq \overline{y}) \mathbf{U} (\overline{c} = \overline{y} \wedge x_{t'})\big)\Big)\Big)
\end{aligned} \tag{3.6}$$

「$\overline{y}$ が不変であるならば，その $\overline{y}$ が示す列のマス目において，その下のマス目でもタイリングが終了しないとき，現マス目と下のマス目とが縦の隣接制約を満たす」ことを表す．ここで,「現マス目と下のマス目とが縦の隣接制約を満たす」を,「現マス目に置くタイル t と，次に再び列が $\overline{y}$ となる最初のマス目，すなわち，同じ列にある直下のマス目に置くタイル t' とが $(t, t') \in V$ を満たす」として記述している.

これにより，$\forall \tilde{y}.\ (\ldots \models (3.6))$ が,「すべての列のおいて，縦の隣接制約を満たす」，つまり,「配置されるすべてのタイルが，縦の隣接制約を満たす」を表すことになる.

定理 3.4.1. *LTL* によるリアクティブシステム仕様の強充足可能性問題は，*EXPSPACE* 困難である.

証明. 上で示したように，EXP-CORRIDOR TILING 問題の答えが「yes」であるとき，かつそのときに限り，強充足可能でない式 φ_{tiling} を構成することができる．φ_{tiling} は，もとの問題に対して多項式サイズであり，多項式時間で構成可能である．すなわち，EXP-CORRIDOR TILING 問題は強充足可能性判定の補問題に多項式帰着可能である．EXP-CORRIDOR TILING 問題は EXPSPACE 完全であるため，強充足可能性判定の補問題は EXPSPACE 困難であり，強充足可能性問題は co-EXPSPACE 困難である．EXPSPACE=co-EXPSPACE であることより，強充足可能性問題は EXPSPACE 困難である． □

3.5 時間演算子のネストの深さを制限した場合の計算量

式の時間演算子のネストの深さを制限した場合の強充足可能性判定問題の計算量について考える．

式 φ の時間演算子のネストの深さ $td(\varphi)$ は，以下のように再帰的に定義される．

- $td(p) = 0$
- $td(\neg\varphi) = td(\varphi)$
- $td(\mathbf{X}\varphi) = td(\varphi) + 1$
- $td(\varphi\mathbf{U}\psi) = max(td(\varphi), td(\psi)) + 1$

また，$td(\varphi)$ を k 以下と制限した LTL のサブセットを LTL(k) とする．

本論文では，時間演算子のネストの深さを 2 以下とした場合の強充足可能性判定問題の計算量について考え，このような制限を与えた場合でも，その計算量は EXPSPACE 完全のままであることを明らかにする．ここでは，LTL の強充足可能性判定問題が，LTL(2) の強充足可能性判定問題へ多項式帰着可能であることを示し，それを証明する．

まず準備として，次のような補題を与える．

補題 3.5.1. *Spec* を要求イベント集合 X と応答イベント集合 Y からなる仕様とする．このとき，任意の振る舞い $\sigma \in (2^{X \cup Y})^\omega$，$\varphi \in sub(Spec)$（*Spec* の部分式）について，

$$\sigma \models Spec \iff \exists \tilde{c}.\langle \sigma, \tilde{c} \rangle \models Spec[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi)$$

である．ここで，$c \notin X \cup Y$ であり，$\tilde{c}$ は $(2^{\{c\}})^\omega$ の要素である．また，$\mathit{Spec}[c/\varphi]$ は Spec 内に出現する φ を c に置き換えた仕様である．

証明． $(\Rightarrow)$: $\sigma \models \mathit{Spec}$ を仮定する．ここで，次のように定義される $\tilde{c}$ を考える．

$$\tilde{c}[i] = \begin{cases} \emptyset & (\sigma[i\ldots] \not\models \varphi \text{ とき}) \\ \{c\} & (\sigma[i\ldots] \models \varphi \text{ とき}) \end{cases}$$

$\tilde{c}$ の定義より，$\langle \sigma, \tilde{c} \rangle \models \mathbf{G}(c \leftrightarrow \varphi)$ である．また，仮定より $\langle \sigma, \tilde{c} \rangle \models \mathit{Spec}$ であり，かつ，任意の i において，$\langle \sigma, \tilde{c} \rangle[i\ldots] \models \varphi \Longleftrightarrow \langle \sigma, \tilde{c} \rangle[i\ldots] \models c$ であるため，$\langle \sigma, \tilde{c} \rangle \models \mathit{Spec}[c/\varphi]$ である．したがって，$\langle \sigma, \tilde{c} \rangle \models \mathit{Spec}[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi)$ である．よって，$\exists \tilde{c}.\langle \sigma, \tilde{c} \rangle \models \mathit{Spec}[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi)$ である．

$(\Leftarrow)$: $\langle \sigma, \tilde{c} \rangle \models \mathit{Spec}[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi)$ を満たす $\tilde{c}$ が存在すると仮定する．$\langle \sigma, \tilde{c} \rangle \models \mathbf{G}(c \leftrightarrow \varphi)$ を満たすため，任意の i において，$\langle \sigma, \tilde{c} \rangle[i\ldots] \models c$ のとき，かつそのときにのみ，$\langle \sigma, \tilde{c} \rangle[i\ldots] \models \varphi$ である．このことと，$\langle \sigma, \tilde{c} \rangle \models \mathit{Spec}[c/\varphi]$ であることより，$\langle \sigma, \tilde{c} \rangle \models \mathit{Spec}$ が導かれる．よって，$\sigma \models \mathit{Spec}$ である． □

補題 3.5.2. Spec を要求イベント集合 X，応答イベント集合 Y からなる仕様とする．Spec が強充足可能であるとき，かつそのときにのみ，要求イベント集合を X，応答イベント集合を $Y \cup \{c\}$ とする仕様 $\mathit{Spec}[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi)$ は，強充足可能である．

証明．

Spec は強充足可能である．

$\Longleftarrow \forall \tilde{a} \exists \tilde{b}.(\langle \tilde{a}, \tilde{b} \rangle \models \mathit{Spec})$

$\Longleftarrow \forall \tilde{a} \exists \tilde{b} \exists \tilde{c}.(\langle \tilde{a}, \langle \tilde{b}, \tilde{c} \rangle \rangle \models \mathit{Spec}[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi))$

（補題 3.5.2 より）

$\Longleftarrow \mathit{Spec}[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi)$ は強充足可能である．

□

これより，強充足可能であるかを変えずに，時間演算子のネストの深さを下げる式の変形が可能であることがわかる．例えば，$\mathit{Spec} = x\mathbf{U}(\mathbf{XFG}y)$ は，

$$\mathit{Spec}_1 = x\mathbf{U}(\mathbf{X}c_{\mathbf{FG}y}) \wedge \mathbf{G}(c_{\mathbf{FG}y} \leftrightarrow \mathbf{FG}y)$$

へ，ネストの深さを減らすよう変換できる（$td(Spec) = 4$，$td(Spec_1) = 3$ である）．

このような変形を繰り返し適用することで，$td(Spec') \leq 2$ であり，かつ強充足可能であるかが $Spec$ と同じであるような $Spec'$ を得られる．これより，LTL(2) における強充足可能性問題が EXPSPACE 完全であることが導かれる．

定理 3.5.1. *LTL(2) によるリアクティブシステム仕様の強充足可能性問題は，EXPSPACE 完全である．*

証明. LTL の強充足可能性判定問題から LTL(2) の強充足可能性判定問題への多項式帰着法を示す．すなわち，LTL による仕様 $Spec$ に対して，$Spec$ が強充足可能であるとき，かつそのときにのみ，強充足可能となる LTL(2) による仕様 $Spec'$ の構成法を示す．$Spec$ を要求イベント集合 X と応答イベント集合 Y からなる LTL 仕様とする．$Spec$ に対して，仕様 $Spec'$ を次のように定義する．

- 要求イベント集合：X.
- 応答イベント集合：$Y \cup C$．ただし，$C = \{c_\varphi \mid \varphi \in sub(Spec)\}$.
- $Spec'$：

$$c_{Spec} \wedge \bigwedge_{c_\varphi \in C} \mathbf{G}(c_\varphi \leftrightarrow \varphi').$$

  ここで，φ' は次のように定める．

  - $\varphi \in X \cup Y$ とき，φ.
  - $\varphi = \neg\psi$ とき，$\neg c_\psi$.
  - $\varphi = \psi \wedge \psi'$ のとき，$c_\psi \wedge c_{\psi'}$.
  - $\varphi = \mathbf{X}\psi$ とき，$\mathbf{X}c_\psi$.
  - $\varphi = \psi\mathbf{U}\psi'$ のとき，$c_\psi\mathbf{U}c_{\psi'}$.

$Spec'$ は，補題 3.5.2 の変形を $Spec$ の各部分式について，大きい式から順に適用させたものとみることができる．したがって，$Spec$ が強充足可能であるとき，かつそのときにのみ，$Spec'$ は強充足可能である．φ' は常に $td(\varphi') \leq 1$ であるため，$Spec'$ は $td(Spec') \leq 2$ である．また，$\mathbf{G}(c_\varphi \leftrightarrow \varphi')$

のサイズは定数であり，$|C| \leq |Spec|$ であるため，$|Spec'| = \mathcal{O}(|Spec|)$ である．

以上のように，LTL の強充足可能性判定問題は，LTL(2) の強充足可能性判定問題へ多項式帰着可能である．前節までで示した通り，LTL の強充足可能性判定問題は EXPSPACE 完全である．したがって，LTL(2) においても強充足可能性判定問題は EXPSPACE 困難であり，その計算量は EXPSPACE 完全となる． □

3.6 議論

充足可能性判定がEXPSPACE完全である時間論理との関係

実時間論理 TPTL[3]，“forgettable past” を扱えるよう拡張された LTL (NLTL) [55]，“freeze quantifier” を扱えるよう拡張された LTL [24] の充足可能性判定は，EXPSPACE 完全であることが知られている．

これらの問題が EXPSPACE 困難であるのは，上の論理においては時刻 i と $i + 2^m$ との間の制約を m に対して多項式サイズの式で表現できるからである．例えば，次の指数サイズの LTL 式によって表現される制約を，それらの論理では多項式サイズで表現できる．

$$\mathbf{G}(\mathbf{XF}(\neg end \wedge \bar{c} = 0) \rightarrow \bigvee_{(t,t') \in V} (x_t \wedge \overbrace{\mathbf{XX}\ldots\mathbf{X}}^{2^m \text{ times}} x_{t'}))$$

この制約は，3.4 節におけるタイリング問題の縦の隣接制約に対応する．この制約は，多項式サイズの（純粋な）LTL 式で表現することができない．本節では，メタレベルの全称限量子 $\forall \tilde{b}$ で限量された LTL 式で，上の制約を表現できることを示し，強充足可能性判定問題の EXPSPACE 困難性を明らかにしている．

他の性質判定問題の計算量との関係

ここでは，充足可能性判定と実現可能性判定の計算量との関係について述べる．

充足可能性判定問題の計算量は PSPACE 完全であること [88]，また，実現可能性判定問題の計算量は 2EXPTIME 完全であること [74] が，こ

れまでに示されている．ここで，PSPACE は決定性チューリング機械で $O(p(n))$ の領域を使って解ける問題のクラス，2EXPTIME は決定性チューリング機械で $O(2^{2^{p(n)}})$ の時間で解ける問題のクラスである．それぞれのクラス間の関係は，以下のようになっている[3]．

$$\mathrm{PSPACE} \subsetneq \mathrm{EXPSPACE} \subseteq \mathrm{2EXPTIME}.$$

したがって，強充足可能性判定問題は，充足可能性判定問題よりも真に難しく，実現可能性判定問題以下の難しさであるといえる．

各種 LTL サブセットの充足可能性判定問題の計算量についての結果は，[25] で示されている．また，LTL サブセットの実現可能性判定問題の計算量については，[4, 69] 等で示されている．特に本章で与えた結果と関係が深いのは，[25] の「LTL の充足可能性判定問題は，時間式のネストの深さを 2 までと制限してもその計算量は PSPACE 完全のままである」という結果である．この結果は，3.5 節で用いた帰着と同様のテクニックを用いて示されている．本研究では，より一般的な強充足可能性という性質を対象としている．本章の補題 3.5.2 は，その帰着テクニックが強充足可能性へ一般化できることを示すものである．

[3] EXPSPACE $\neq$ 2EXPTIME が成立するかは明らかになっていないが，P と NP の関係と同様，多くの研究者は真部分集合の関係にあると考えている．

第4章　有界強充足可能性判定

本章では，強充足可能性を基にした有界検査手法を与える．この手法は，リアクティブシステム仕様の単純な欠陥を効率的に検出することを目的としている．

前章でも述べたように，強充足可能性は，実現可能性よりも少ない計算コストで判定できる．また，実現可能性の必要条件ではあるが，実用的なリアクティブシステム仕様の多くが実現不能ならば強充足不能でもあるため，強充足可能性を基にする検証で多くの欠陥を検出できる．

本章では，この強充足可能性をさらに制限した有界強充足可能性を導入する．ここで導入する有界強充足可能性は，強充足可能性をサイズ k の繰り返し構造として表現される要求イベント集合列のみを考慮するように制限した性質である．つまり，仕様, パラメータ k に対して,「サイズ k の繰り返し構造として表現される任意の要求イベント集合列に対して，仕様を満たす応答イベント集合列が存在する」という性質であり，単純な要求パターンに対する応答可能性を表す性質であるといえる．強充足不能な仕様には比較的小さな反例（仕様を満たすように応答不能なサイズ k の繰り返し構造で表される要求パターン）が存在することが多い．それゆえ，この性質の検査によりリアクティブシステム仕様の多くの欠陥を検出できると予想される．

ここでは，この有界強充足可能性が満たされるかを SAT ソルバを用いて判定する手法を与える．SAT ソルバとは，命題論理式の充足可能性判定を行うツールのことである．命題論理の充足可能性判定問題は NP 完全であるが，これまでに様々な効率化技法が提案されており（[87, 98] 等），近年の SAT ソルバは驚くほど高速である．代表的な SAT ソルバとしては，zChaff[66]， MiniSAT[28]，PicoSAT[13] 等がある．

本判定法では，まず，仕様を満たすように応答可能な要求イベント集合列をちょうど受理する非決定性 Büchi オートマトン（NBA）を構成する．そして，そのオートマトンがサイズ k の繰り返し構造として表現されるすべての要求イベント集合列を受理する（k-全受理である）かどう

かを調べる．本研究では，この NBA の k-全受理判定を SAT ソルバを用いて行う．

本研究では，LTL で記述されたリアクティブシステム仕様の有界強充足可能性問題の計算量についても考察し，それが co-NEXPTIME 完全であることを示す．前章で示したように，(非有界）強充足可能性判定問題は EXPSPACE 完全であり，実現可能性判定問題は 2EXPTIME 完全である．したがって，ここで示す結果により，有界強充足可能性問題は，強充足可能性判定問題や実現可能性判定問題以下の難しさであることが明らかになる．

4.1 有界強充足可能性

本章で導入する有界強充足可能性は，強充足可能性をサイズ k の繰り返し構造として表現される要求イベント集合列のみを考慮するように制限した性質である．すなわち,「任意のサイズ k の繰り返し構造として表現される要求イベント集合列に対して，仕様を満たす応答イベント集合列が存在する」という性質であり，次のように定義する．

定義 4.1.1 (k-ループ)**.** $l \leq k$ とし，σ を無限列とする．σ に対して $\sigma = u{\cdot}v^{\omega}$ であるような $u = s_0 s_1 \ldots s_{l-1}$，$v = s_l s_{l+1} \ldots s_k$ が存在するとき，σ を (k,l)-ループと呼ぶ．また， σ が (k,l)-ループとなるような $l \leq k$ が存在するとき，σ を単に k-ループと呼ぶ.

定義 4.1.2 (k-強充足可能性)**.** *Spec* を仕様，$k \in \mathbb{N}$ とする．*Spec* が，任意の k-ループである要求イベント集合列 $\tilde{a}$ について，

$$\exists \tilde{b}.(\langle \tilde{a}, \tilde{b} \rangle \models \mathit{Spec})$$

を満たすとき，*Spec* は k-強充足可能であるという．

上の定義では，考慮する要求イベント集合列をサイズ k のループとして表現されるものに制限しているが，これは，サイズ k 以下のループに制限することと同値である．すなわち，$k < k'$ であるとき，*Spec* が k'-強充足可能であるならば，k-強充足可能でもある．

定理 4.1.1. σ を無限列とし，$k < k'$ とする．このとき，σ が k-ループであるならば，σ は k'-ループでもある．

証明. σ が k-ループであるならば，σ が $(k+1)$-ループでもあることを示せば十分である．σ が k-ループであるとする．すなわち，ある $l \leq k$ が存在して $\sigma = s_0 s_1 \ldots s_{l-1}(s_l s_{l+1} \ldots s_k)^\omega$ とする．このとき，$\sigma = s_0 s_1 \ldots s_{l-1} s_l (s_{l+1} \ldots s_k s_l)^\omega$ であるため，σ は $(k+1)$-ループでもある． □

定理 4.1.2. *Spec* を仕様とし，$k < k'$ とする．*Spec* が k'-強充足可能であるならば，k-強充足可能でもある．

証明. 定理 4.1.1 より． □

有界強充足可能性と（非有界な）強充足可能性との関係について述べる．定義から明らかなように，*Spec* が強充足可能であるならば k-強充足可能でもある．仕様が LTL で記述される場合（非決定性 Büchi オートマトンでそれを表現できる場合），その逆も k を十分に大きくしたとき成立する．つまり，k を十分に大きくすると，k-強充足可能は強充足可能性と同等となる．

定理 4.1.3. *Spec* を仕様，$k \in \mathbb{N}$ とする．*Spec* が強充足可能であるならば，*Spec* は k-強充足可能でもある．

定理 4.1.4. *Spec* を *LTL* によって記述される仕様とする．このとき，*Spec* が強充足不能であるならば，ある $k \in \mathbb{N}$ が存在して，*Spec* が k-強充足不能である．

証明. *Spec* を満たすよう応答可能な要求イベント集合列をちょうど受理する，つまり $L(\mathcal{A}') = \{\tilde{a} \mid \exists \tilde{b}.\langle \tilde{a}, \tilde{b} \rangle \models \mathit{Spec}\}$ となる非決定性 Büchi オートマトン $\mathcal{A}'$ を *Spec* より構成することが可能である（その構成方法については 3.3 節を参照されたい）．非決定性 Büchi オートマトンによって認識可能な言語のクラスは補集合演算について閉じているため，*Spec* を満たすよう応答不能な要求イベント集合列をちょうど受理する，つまり $L(\overline{\mathcal{A}'}) = \{\tilde{a} \mid \forall \tilde{b}.\langle \tilde{a}, \tilde{b} \rangle \not\models \mathit{Spec}\}$ である非決定性 Büchi オートマトン $\overline{\mathcal{A}'}$ が存在する．*Spec* が強充足不能であると仮定すると，$\overline{\mathcal{A}'}$ に受理される語が存在することになる．非決定性 Büchi オートマトンが非空であるならば，そのオートマトンに受理される $\sigma = u \cdot v^\omega$（ここで，$u, v$ は有限長の語）が存在する [91]．それゆえ，*Spec* を満たすように応答不能な k-ループである要求イベント集合列が存在する．よって，*Spec* が強充足不能であるならば，ある $k \in \mathbb{N}$ が存在して，*Spec* は k-強充足不能である． □

例として，2.3 節と 3.1 節で扱ったドア制御システムの仕様

$$door : \mathbf{G}((x_1 \to \mathbf{F}y) \wedge (x_2 \to \neg y))$$

について考える．3.1 節でも述べたように，要求イベント集合列 $\tilde{a} = \{x_1, x_2\}\{x_2\}\{x_2\}\ldots$ に対して，*door* を満たす応答イベント集合列は存在しない，すなわち，$\exists\tilde{b}\langle\tilde{a}, \tilde{b}\rangle \models door$ は満みたされない．この要求イベント列集合は，$\tilde{a} = \{x_1, x_2\}\{x_2\}^\omega$ と表現できるため，1-ループである．それゆえ，仕様 *door* は 1-強充足可能性を満たさない．

このように有界強充足可能性は，リアクティブシステム仕様の単純な要求パターンに対する応答可能性を表す性質である．この性質を調べることで，仕様を満たすよう応答不能な単純な要求パターンが存在するという欠陥があるかを検査することができる．強充足不能な仕様は，小さなループで表現される比較的単純な反例（仕様を満たす応答が存在しない要求パターン）を持つことが多い．それゆえ，このような限られた範囲の応答可能性に関する検査でも，実際的なリアクティブシステム仕様の多くの欠陥を見つけられることが予想される．

4.2 有界強充足可能性判定

本節では，3.3 節の NBA を用いた（非有界）強充足可能性判定手続きを基に，有界強充足可能性手続きを定義する．

Spec を LTL によるリアクティブシステム仕様とし，k をバウンドとする．*Spec* の k-強充足可能性の判定は，次の手順で行うことができる．

1. *Spec* を満たす振る舞いをちょうど受理する，つまり，$L(\mathcal{A}) = \{\sigma \mid \sigma \models Spec\}$ となる NBA $\mathcal{A} = \langle 2^{X \cup Y}, Q, q_I, \delta, F\rangle$ を構成する．

2. $\mathcal{A}$ をもとに，アルファベットを要求イベントのみに制限した NBA $\mathcal{A}' = \langle 2^X, Q, q_I, \delta', F\rangle$ を構成する．ここで，$\delta' = \{(q, a, q') \mid \exists b.(q, a \cup b, q') \in \delta\}$ である．

   - $L(A') = \{\tilde{a} \mid \exists\tilde{b}.\langle\tilde{a}, \tilde{b}\rangle \models Spec\}$ となる．

3. $\mathcal{A}'$ が任意の k-ループを受理するか（以降，k-全受理であるという）を調べる．k-全受理であるならば,「k-強充足可能」を出力．受理されない k-ループ $\tilde{a}$ が存在する場合,「k-強充足可能でない」を出力する．

前章で与えた非有界な強充足可能性判定手続きとの違いは，最終ステップにある．強充足可能性判定手続きでは，$\mathcal{A}'$ が全受理であるか（任意の語を受理するか）を調べることで，強充足可能を満たすかを判定する．それに対して，k-強充足可能性の検査では，$\mathcal{A}'$ が k-全受理であるか（任意の k-ループを受理するか）を調べることで，k-強充足可能であるかを判定する．

本研究では，SAT ソルバを用いてこの k-全受理判定を行う．次節でその判定法を与える．

4.3 SAT ソルバを用いた NBA の k-全受理判定

本章では，SAT ソルバを用いた NBA の k-全受理判定法を与える．ここで与える k-受理判定法では，k-全受理判定の補問題を SAT 問題に帰着する．すなわち，NBA $\mathcal{A}$, バウンド k に対して，$\mathcal{A}$ に受理されない k-ループが存在するとき，かつそのときにのみ，充足可能となる命題論理式 f を用意し，その SAT 判定を行うことで，k-全受理でないかどうかを調べる．

4.3.1 k-loop が不受理であることの特徴付け

NBA に受理されない k-ループが存在するとき，かつそのときにのみ，充足可能となる命題論理式を定義する準備として，k-ループが NBA に受理されない条件について考察する．

$\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ を NBA とし，σ を (k, l)-ループとする．σ が $\mathcal{A}$ に受理されないのは，$\mathcal{A}$ に σ の受理行程が存在しないときである．すなわち，σ についてのすべての行程 ϱ において $inf(\varrho) \cap F = \emptyset$ である（受理状態が有限回のみ出現する）ときである．ここでは次に定義するようなすべての行程を表現する行程グラフを基に，(k, l)-ループが不受理となる条件について考える．

定義 4.3.1 (行程グラフ)**.** *NBA* $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$，$(k, l)$-ループ $\sigma = s_0 \ldots s_{l-1}(s_l \ldots s_k)^\omega$ に対して，行程グラフ $G = \langle V, v_I, E, C \rangle$ を以下のように定義する．

- （ノード集合）$V := Q \times \{0, 1, \ldots, k\}$
- （初期ノード）$v_I := (q_I, 0)$

- （エッジ集合）
  $E := \{((q,i),(q',suc(i))) \mid (q,s_i,q') \in \delta\}$
  ここで $suc(i)$ は，$i = k$ とき l を，その他のとき $i+1$ を表す.

- （受理ノード）$C := \{(q,i) \mid q \in F, 0 \leq i \leq k\}$

例として，図 4.1 で示すような NBA $\mathcal{A}_{ex}$ について考える（q_0 が初期状態であり，二重円の q_1 は受理状態である）．この $\mathcal{A}_{ex}$ と $(1,0)$-ループ $\sigma_{ex} = (ab)^\omega$ についての行程グラフは，図 4.2 のようになる（二重四角で表わされるノードが受理ノードである）．

行程グラフの (q,i) からのパスは，$\sigma[i\ldots]$ に対する $\mathcal{A}$ の q からの行程と対応し，対応する行程が受理であるとき，かつそのときのみ，受理ノードを無限回通過する．したがって，以下が成立する．

定理 4.3.1. *$\mathcal{A}$ を NBA とし，σ を (k,l)-ループとする．σ が $\mathcal{A}$ に受理されないならば，かつそのときに限り，$\mathcal{A}$, σ の行程グラフの初期ノードからの任意のパスにおいて，受理ノードの出現は有限回である.*

ここで考える行程グラフのサイズが有界であるため,「行程グラフの初期ノードからの任意のパスにおいて，受理ノードの出現が有限回のみ」と「初期ノードからの任意のパスにおいて，受理ノードの出現は d 回以下」とが同等なる有界な d が存在する．したがって，上の不受理の条件は有界なものへと変更できる.

補題 4.3.1. *$G = \langle V, v_I, E, C\rangle$ を行程グラフとし，$d \geq |C|$ とする．このとき，行程グラフ G の初期ノードからの任意のパスにおいて，受理ノードの出現が有限回のみであるならば，G の初期ノードからの任意のパスにおいて，受理ノードの出現は d 回以下である.*

証明. 対偶を示す．G に，受理ノードの出現が d 回より多い，初期ノードからのパス $\tilde{v}$ が存在すると仮定する．$d \geq |C|$ であるため，$\tilde{v}$ には，ある $v_c \in C$ が 2 回以上出現することになる．これは，v_c から再び v_c へ到るパスが存在することを意味し，これより，初期ノードから繰り返し v_c を通過するパス $\tilde{v}'$ の存在が導かれる．$\tilde{v}'$ では受理ノードが無限回出現するので，受理ノードの出現が無限回となる初期ノードからのパスが存在することになる．したがって，題意は満たされる． □

定理 4.3.2. *$\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F\rangle$ を NBA，$k \in \mathbb{N}$ とする．このとき，任意の $d \in \mathbb{N}$ で，以下の (2) $\Rightarrow$ (1) が成立し，$d \geq (k+1)\cdot|F|$ のとき，(1) $\Rightarrow$ (2) も成立する.*

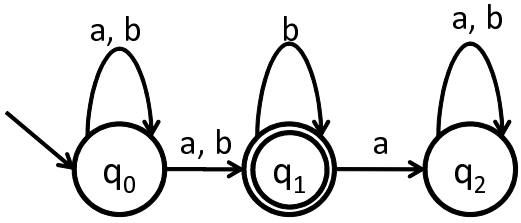

図 4.1: NBA $\mathcal{A}_{ex}$

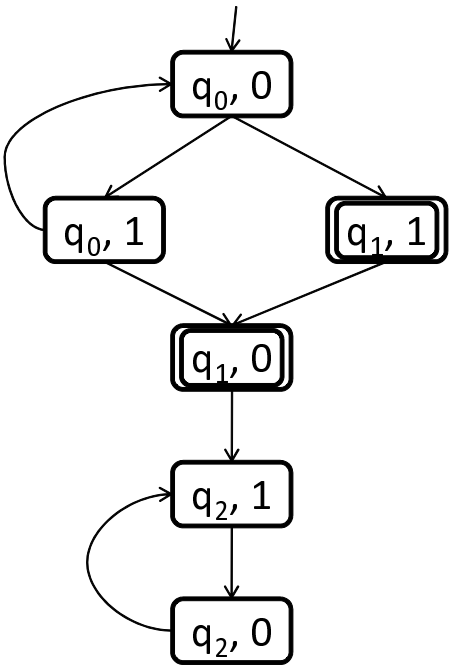

図 4.2: $\mathcal{A}_{ex}$, σ_{ex} の行程グラフ

1. $\mathcal{A}$ に受理されない k-ループが存在する．

2. ある k-ループ σ が存在し，$\mathcal{A}$, σ に対する行程グラフの初期ノードからの任意のパスにおいて，受理ノードの出現が d 回以下である．

証明. 定理 4.3.1 と補題 4.3.1 より． □

k-ループ σ が NBA $\mathcal{A}$ に受理されないとは，$d \geq (k+1) \cdot |F|$ における「$\mathcal{A}$, σ に対する行程グラフの初期ノードからの任意のパスにおいて，受理ノードの出現が d 回以下である」と同等であることがかわかった．命題論理により表現するため，この条件についてさらにみていく．

「行程グラフ $G = \langle V, v_I, E, C \rangle$ のノード v からの任意のパスにおいて，受理ノードの出現が j 回以下である」ことは，以下のように特徴付けられる．

- $v \in V \setminus C$ のとき，
  「行程グラフの v からのすべてのパスで，受理ノードの出現が j 回以下である」**iff** v のすべての遷移先 v' で,「v' からのすべてのパスで，受理ノードの出現が j 回以下である」

- $v \in C$ のとき，
  $j > 0$ ならば,「行程グラフの v からのすべてのパスで，受理ノードの出現が j 回以下である」**iff** v のすべての遷移先 v' で,「v' からのすべてのパスで，受理ノードの出現が $j-1$ 回以下である」．
  「行程グラフの v からのすべてのパスで，受理ノードの出現が 0 回以下である」ことはない．

これを基に，以下が成立する．

定理 4.3.3. $G_\sigma^{\mathcal{A}} = \langle V, v_I, E, C \rangle$ を $\mathcal{A}, \sigma$ についての行程グラフとする．行程グラフ $G_\sigma^{\mathcal{A}}$ の初期ノードからの任意のパスにおいて受理ノードの出現が d 回以下であるならば，かつそのときに限り，以下を満たすノード集合 $V_0, V_1, \ldots, V_d$ が存在する．

1. V_0 が以下の条件を満たす（以降，$I_\sigma^{\mathcal{A}}(V_0)$ と表記する）．

$$v \in V_0 \iff \begin{cases} \forall v' \in vE.\ v' \in V_0 & (v \in V \setminus C \text{ のとき}) \\ \bot & (v \in C \text{ のとき}) \end{cases}$$

ここで $vE := \{v' \in V \mid (v, v') \in E\}$ とする．

2. 任意の $0 \leq j < d$ において，以下の条件（以降，$T_\sigma^{\mathcal{A}}(V_j, V_{j+1})$ と表記）を満たす.

$$v \in V_{j+1} \iff \begin{cases} \forall v' \in vE.\ v' \in V_{j+1} & (v \in V \setminus C \text{ のとき}) \\ \forall v' \in vE.\ v' \in V_j & (v \in C \text{ のとき}) \end{cases}$$

3. V_d が $v_I \in V_d$ を満たす（以降，この条件を $F_\sigma^{\mathcal{A}}(V_d)$ と表記する）.

証明. （$\Longrightarrow$）行程グラフ $G_\sigma^{\mathcal{A}}$ の初期ノードからの任意のパスにおいて受理ノードの出現が d 回以下であると仮定する．そこからの任意のパスにおいて受理ノードの出現が j 回以下であるノードの集合を V_j する．このとき，$V_0, \ldots, V_d$ は，$I_\sigma^{\mathcal{A}}(V_0) \wedge \bigwedge_{0 \leq j < d} T_\sigma^{\mathcal{A}}(V_j, V_{j+1}) \wedge F_\sigma^{\mathcal{A}}(V_d)$ を満たす.

（$\Longleftarrow$）$I_\sigma^{\mathcal{A}}(V_0) \wedge \bigwedge_{0 \leq j < d} T_\sigma^{\mathcal{A}}(V_j, V_{j+1}) \wedge F_\sigma^{\mathcal{A}}(V_d)$ を満たすノード集合 $V_0, V_1, \ldots, V_d$ が存在し，かつ行程グラフ $G_\sigma^{\mathcal{A}}$ に受理ノードの出現が $d+1$ 回以上である初期ノードからのパス $\tilde{v} = v_0 v_1 \ldots$ が存在すると仮定し，矛盾を導く．$\tilde{v}$ において受理ノードである最初の $d+1$ 個の要素がそれぞれ $i_d, i_{d-1}, \ldots, i_0$ 番目の要素であるとする（$i_d \leq i_{d-1} \leq \ldots \leq i_0$ とする）．$F_\sigma^{\mathcal{A}}(V_d)$ であることより $v_0 \in V_d$ でなればならない．また，$\bigwedge_{0 \leq j < d} T_\sigma^{\mathcal{A}}(V_j, V_{j+1})$ であることより，すべての $0 \leq i \leq i_d$ で $v_i \in V_d$ でなければならず，同様に，任意の $d > j \geq 0$ のすべての $i_{j+1} < i \leq i_j$ で $v_i \in V_j$ でなけらばならない．これより $v_{i_0} \in V_0$ でなければならないが，これは，$I_\sigma^{\mathcal{A}}(V_0)$ を満たすこと，および，$v_{i_0} \in C$ であることと矛盾する． □

先の図 4.2 の行程グラフを例に考える．この行程グラフでは,「初期ノードからの任意のパスにおいて受理ノードの出現は 2 回以下」であり，上の条件を満たすようなノード集合 V_0, V_1, V_2 が存在する.

$$\begin{aligned} V_0 &= \{(q_2, 0), (q_2, 1)\} \\ V_1 &= \{(q_1, 0), (q_2, 0), (q_2, 1)\} \\ V_2 &= \{(q_0, 0), (q2, 0), (q_1, 1), (q_1, 0), (q_2, 0), (q_2, 1)\} \end{aligned}$$

一方で,「初期ノードからの任意のパスにおいて受理ノードの出現は 1 回以下」ではなく（初期ノードからの受理ノードが 2 回出現するパス $\tilde{v} = (q_0, 0)(q_1, 1)(q_1, 0)(q_2, 1) \ldots$ が存在），上の条件を満たすようなノード集合 V_0, V_1 は存在しない.

これまでの考察をまとめると，以下のようになる.

定理 4.3.4. $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ を NBA，$k \in \mathbb{N}$ とする．このとき，任意の $d \in \mathbb{N}$ で，以下の (2) $\Rightarrow$ (1) が成立し，$d \geq (k+1) \cdot |F|$ のとき，(1) $\Rightarrow$ (2) も成立する．

1. $\mathcal{A}$ に受理されない k-ループが存在する．
2. ある k-ループ σ が存在し，$I_\sigma^{\mathcal{A}}(V_0) \wedge \bigwedge_{0 \leq j < d} T_\sigma^{\mathcal{A}}(V_j, V_{j+1}) \wedge F_\sigma^{\mathcal{A}}(V_d)$ を満たすような $\mathcal{A}$, σ の行程グラフ $G_\sigma^{\mathcal{A}}$ のノード集合列 $V_0, V_1, \ldots, V_d$ が存在する．

以降において，条件 (2) を満たすことを $notAcc(\mathcal{A}, k, d)$ と示す．

4.3.2 SAT 問題への帰着

$\mathcal{A}$ に受理されない k-ループの存在判定を，前節の定理 4.3.4 を基に SAT 問題へ帰着する方法を与える．すなわち，$notAcc(\mathcal{A}, k, d)$（定理 4.3.4 の条件 2）を満たすとき，かつそのときにのみ，充足可能となる命題論理式 $|[notAcc(\mathcal{A}, k, d)]|$ の構成方法を与える．

ここで与える帰着では，それらの真偽割り当てで 1 つの k-ループ，及び $V_0, V_1, \ldots, V_d$ を表すよう命題変数を設け,「正しく k-ループである」こと，及び $I_\sigma^{\mathcal{A}}(V_0) \wedge \bigwedge_{0 \leq j < d} T_\sigma^{\mathcal{A}}(V_j, V_{j+1}) \wedge F_\sigma^{\mathcal{A}}(V_d)$ を表現する命題論理式を構成する．

命題変数

それらの真偽で，ひとつの (k, l)-ループを表すよう，以下の命題変数を準備する．ただし，アルファベットは $\Sigma = 2^P$ であり，(k, l)-ループの各要素は P の部分集合であると仮定する．

- p_i: $p \in P, 0 \leq i \leq k$.
  k-ループの i 番目の要素に p が含まれているかを表す．
- l_i: $0 \leq i \leq k$.
  k 番目から i 番目へ戻りが存在するかを表す．

また，$V_0, V_1, \ldots, V_d$ を表すため，以下の変数を導入する．

- $v_{(q,i)}^j$: $q \in Q, 0 \leq i \leq k, 0 \leq j \leq d$.
  $(q, i) \in V_j$ であるかを表す．

$\lvert[I(\mathcal{A},k)]\rvert_0$	$\bigwedge_{\substack{q\in Q\setminus F,\\ 0\le i\le k}} \left(v^0_{(q,i)} \leftrightarrow \bigwedge_{(q,a,q')\in\delta} (\lvert[a]\rvert_i \to \lvert[suc]\rvert^0_{(q',i)}) \right) \wedge \bigwedge_{\substack{q\in F,\\ 0\le i\le k}} (\neg v^0_{(q,i)})$
$\lvert[T(\mathcal{A},k)]\rvert_{j,j+1}$	$\bigwedge_{\substack{q\in Q\setminus F,\\ 0\le i\le k}} \left(v^{j+1}_{(q,i)} \leftrightarrow \bigwedge_{(q,a,q')\in\delta} (\lvert[a]\rvert_i \to \lvert[suc]\rvert^{j+1}_{(q',i)}) \right) \wedge \bigwedge_{\substack{q\in F,\\ 0\le i\le k}} \left(v^{j+1}_{(q,i)} \leftrightarrow \bigwedge_{(q,a,q')\in\delta} (\lvert[a]\rvert_i \to \lvert[suc]\rvert^{j}_{(q',i)}) \right)$
$\lvert[F(\mathcal{A},k)]\rvert_d$	$v^d_{(q_I,0)}$

表 4.1: $I^{\mathcal{A}}_\sigma(V_0)$, $T^{\mathcal{A}}_\sigma(V_i,V_{i+1})$ 及び $F^{\mathcal{A}}_\sigma(V_d)$ の命題論理式による表現

ループの制約

「正しく k-ループである」，つまり「k 番目の要素から，ただひとつの要素へのみ戻りが存在する」という制約 $\lvert[loop(k)]\rvert$ は，次のように定義される．

$$\lvert[loop(k)]\rvert := \bigvee_{0\le i\le k} l_i \wedge \bigwedge_{0\le i\le k} (l_i \to \bigwedge_{\substack{0\le i'\le k,\\ i'\ne i}} \neg l_{i'})$$

不受理であることの表現

続いて，$V_0, V_1, \ldots, V_d$ が $I^{\mathcal{A}}_\sigma(V_0) \wedge \bigwedge_{0\le j<d} T^{\mathcal{A}}_\sigma(V_j,V_{j+1}) \wedge F^{\mathcal{A}}_\sigma(V_d)$ を満たすことを表現する命題論理式の構成法について述べる．

$I^{\mathcal{A}}_\sigma(V_0)$, $T^{\mathcal{A}}_\sigma(V_i,V_{i+1})$ 及び $F^{\mathcal{A}}_\sigma(V_d)$ であることを表す命題論理式をそれぞれ定義していく．それぞれを $\lvert[I(\mathcal{A},k)]\rvert_0$, $\lvert[T(\mathcal{A},k)]\rvert_{j,j+1}$, $\lvert[F(\mathcal{A},k)]\rvert_d$ で表す．NBA $\mathcal{A} = \langle 2^P, Q, q_I, \delta, F\rangle$ とすると，$\lvert[I(\mathcal{A},k)]\rvert_0$, $\lvert[T(\mathcal{A},k)]\rvert_{j,j+1}$ 及び $\lvert[F(\mathcal{A},k)]\rvert_d$ は表 4.1 のように定義される．ここで，$\lvert[a]\rvert_i$ は，k-ループの i 番目の要素が a であることを表す式であり，以下のように定義される．

$$\lvert[a]\rvert_i := \bigwedge_{p\in a} p_i \ \wedge \ \bigwedge_{p\notin a} \neg p_i$$

また，$|[suc]|^j_{(q,i)}$ は，$(q, suc(i)) \in V_j$ であることを表す式であり，以下のように定義される．

$$|[suc]|^j_{(q,i)} := \begin{cases} \bigwedge_{0 \le i' \le k} l_{i'} \to v^j_{(q,i')} & (i = k \text{ のとき}) \\ v^j_{(q,i+1)} & (\text{その他のとき}) \end{cases}$$

これまでに与えた各論理式を用い，命題論理式 $|[notAcc(\mathcal{A}, k, d)]|$ は以下のように定義できる．

$$|[loop(k)]| \ \wedge \ |[I(\mathcal{A}, k)]|_0 \ \wedge \ \bigwedge_{0 \le i < d} |[T(\mathcal{A}, k)]|_{j,j+1} \ \wedge \ |[F(\mathcal{A}, k)]|_d$$

定理 4.3.5. $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ を *NBA*，$k \in \mathbb{N}$ とする．このとき，任意の $d \in \mathbb{N}$ で，以下の (2) $\Rightarrow$ (1) が成立し，$d \ge (k+1) \cdot |F|$ のとき，(1) $\Rightarrow$ (2) も成立する．

1. $\mathcal{A}$ に受理されない k-ループが存在する．
2. $|[notAcc(\mathcal{A}, k, d)]|$ が充足可能である．

証明. 定理 4.3.4 より． □

定理 4.3.6. *NBA* $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$，$k, d \in \mathbb{N}$ とする．このとき，$|[notAcc(\mathcal{A}, k, d)]|$ のサイズは，$\mathcal{O}(k^2 + k \cdot d \cdot |\delta|)$ である．

証明. $|[loop(k)]|$, $|[I(\mathcal{A}, k)]|_0$, $|[T(\mathcal{A}, k)]|_{j,j+1}$, $|[F(\mathcal{A}, k)]|_d$ のサイズが，それぞれ $\mathcal{O}(k^2)$, $\mathcal{O}(k \cdot |\delta|)$, $\mathcal{O}(k \cdot |\delta|), \mathcal{O}(1)$ であることによる． □

定理 4.3.7. *NBA* $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$，$k \in \mathbb{N}$ とする．このとき，$\mathcal{A}$ が k-全受理でないかの判定は，$\mathcal{O}(k^2 \cdot |F| \cdot |\delta|)$ のサイズの命題論理式の *SAT* 判定に帰着可能である．

4.3.3 インクリメンタルな k-全受理判定法

前章にて，$d \ge (k+1) \cdot |F|$ での $notAcc(\mathcal{A}, k, d)$ を調べることで，NBA $\mathcal{A}$ が k-全受理でないかを判定できることを示した．本章では，d の値を段階的に増進させて k-全受理であるかを調べる，インクリメンタルな判定法について述べる．

インクリメンタルな k-全受理判定法

定理 4.3.4 で示したように，$d < (k+1) \cdot |F|$ のときでも（4.3.4 節の帰着法では $d < |Q|$ のときでも）$notAcc(\mathcal{A}, k, d)$ が満たされることが分かれば,「k-全受理でない」と判定できる．したがって，NBA $\mathcal{A}$ の k-全受理の判定は以下のようにも行うことができる．

定義 4.3.2 (ナイーブなインクリメンタルな k-全受理判定).
入力：*NBA* $\mathcal{A}$, バウンド k.
出力：「k-全受理である」 *or*「k-全受理でない」

1. $d := 0$.
2. $notAcc(\mathcal{A}, k, d)$ であるかを調べる．
   - $notAcc(\mathcal{A}, k, d)$ が満たされるならば,「k-全受理でない」と判定.
   - そうでないならば，次へ.
3. $d \geq (k+1) \cdot |F|$ であるか？
   - そうであるならば「k-全受理である」と判定.
   - そうでないならば，次へ.
4. $d := d + 1$ として，*2* へ.

この判定法は，NBA に受理されない k-ループが存在する多くの場合，$d = (k+1) \cdot |F|$ での $notAcc(\mathcal{A}, k, d)$ だけを調べる方法よりも，効率的に「k-全受理でない」と判定できる．前章で述べたように，$notAcc(\mathcal{A}, k, d)$ を調べるのに用いる命題論理式のサイズは $\mathcal{O}(k^2 + k \cdot d \cdot |\delta|)$ であり，d が小さいときの充足可能性判定のコストは，d が大きい場合に比べて低い．また，NBA が k-全受理でないとき，$(k+1) \cdot |F|$ に比べて小さな d で，$notAcc(\mathcal{A}, k, d)$ を満たす（「$\mathcal{A}$，σ による行程グラフの初期ノードからのすべてのパスで，受理ノードの出現が d 回以下」である k-ループ σ が存在する）ことが多い．そのため，NBA に受理されない k-ループが存在する場合，そのような k-ループを効率的に発見できる．

このような判定法では，式の充足可能性を繰り返し調べることになるが，これはインクリメンタル SAT ソルバ（MiniSAT[28] 等）を用いて効率的に行える．インクリメンタル SAT ソルバは，式 f の充足可能性判定を行った後，f と形が似た式 f' の充足可能性判定を行うとき，f の判定で得られた

情報を用い，f' の充足可能性判定を効率的に行える．式 $|[notAcc(\mathcal{A}, k, j)]|$ と式 $|[notAcc(\mathcal{A}, k, j+1)]|$ の形は近いため，上のインクリメンタルな k-全受理判定法は，このような SAT ソルバとの相性がよい．

インダクションを用いたインクリメンタルな k-全受理判定法

上で述べた判定法では，「k-全受理である」と判定するためには，$d \geq (k+1) \cdot |F|$ での $notAcc(\mathcal{A}, k, d)$ が満たされないことを示す必要がある．d が大きいほど扱う命題論理式のサイズが大きくなるため，「k-全受理である」と判定するコストは，「k-全受理でない」ことを示す場合に比べて高い．ここでは，この問題を軽減させる方法について述べる．すなわち，各 d での $notAcc(\mathcal{A}, k, d)$ の検査後，その段階で k-全受理であるとジャッジできるかを $notAcc(\mathcal{A}, k, d)$ の検査同様 SAT ソルバを用いて検査する判定法について述べる．ここで述べる判定法は，[77, 10] 等で提案された SAT ベースの到達可能性検査法で用いられているインダクションのアイディアを基する．

ここで与える判定法では，$notAcc(\mathcal{A}, k, d)$ が満たされないことがわかった後，以下（以降，$induction(\mathcal{A}, k, d)$ と示す）が成立するかを調べる．

$$\exists\sigma.\exists V_0, V_1, \ldots V_{d+1}.$$
$$I_\sigma^{\mathcal{A}}(V_0) \wedge \bigwedge_{0 \leq j < d+1} T_\sigma^{\mathcal{A}}(V_j, V_{j+1}) \wedge \bigwedge_{0 \leq j < d+1} V_j \subseteq V_{j+1} \wedge \bigwedge_{0 \leq j < d+1} V_j \neq V_{j+1}.$$

この条件が満たされない場合，その段階で任意の n において $notAcc(\mathcal{A}, k, n)$ でない，つまり $\mathcal{A}$ が k-全受理であることが導かれる．

直感的には $induction(\mathcal{A}, k, d)$ は，「ある σ における行程グラフにおいて，$I_\sigma^{\mathcal{A}}$ と $T_\sigma^{\mathcal{A}}$ を満たし，かつ，同じ集合を含まない，昇順に並らんだノード集合列 $V_0, V_1, \ldots V_{d+1}$ が存在する」ことを表す．定義より，$induction(\mathcal{A}, k, d)$ が成立しないならば，任意の $n > d$ においても $induction(\mathcal{A}, k, n)$ は成立しない．また，定理 4.3.3 の条件に「ノード集合列 $V_0, V_1, \ldots V_d$ は，同じ集合を含まず，昇順に並ぶ」という条件を足しても，定理 4.3.3 の性質は成立する．これらより，$notAcc(\mathcal{A}, k, d)$ が満たされず，$induction(\mathcal{A}, k, d)$ も満たされないならば，任意の n においても $notAcc(\mathcal{A}, k, n)$ は満たされないことが導かれる．

定理 4.3.8. *NBA* $\mathcal{A}$，バウンド k とし，$d \in \mathbb{N}$ とする．$notAcc(\mathcal{A}, k, d)$ を満たさず，$induction(\mathcal{A}, k, d)$ も満たさないならば，任意の n においても $notAcc(\mathcal{A}, k, n)$ を満たさない．

証明. $\neg induction(\mathcal{A}, k, d)$ と $\neg notAcc(\mathcal{A}, k, n)$ を仮定して，$\forall n.\neg notAcc(\mathcal{A}, k, n)$ を導く．

まず，仮定から $\forall n.\neg notAcc'(\mathcal{A}, k, n)$ を導く．ここで， $notAcc'(\mathcal{A}, k, n)$ は

$$\begin{aligned}&\exists\sigma.\exists V_0, V_1, \ldots V_n.\\&\quad I_\sigma^{\mathcal{A}}(V_0) \wedge \bigwedge_{0\le j<n} T_\sigma^{\mathcal{A}}(V_j, V_{j+1}) \wedge F_\sigma^{\mathcal{A}}(V_n) \wedge\\&\quad \bigwedge_{0\le j<n} V_j \subseteq V_{j+1} \wedge \bigwedge_{0\le j<n} V_j \neq V_{j+1}\end{aligned}$$

である．$induction(\mathcal{A}, k, d)$ の定義より，$\neg induction(\mathcal{A}, k, d)$ ならば，任意の $n > d$ においても $\neg induction(\mathcal{A}, k, n)$ である．これと $induction(\mathcal{A}, k, n)$ の定義から，$\forall n \ge d+1.\neg notAcc'(\mathcal{A}, k, n)$ が得られる．また，定理 4.3.3 より，$\neg notAcc(\mathcal{A}, k, d)$ ならば，$\forall n \le d.\neg notAcc(\mathcal{A}, k, n)$ である．これと $notAcc'(\mathcal{A}, k, n)$ の定義から，$\forall n \le d.\neg notAcc'(\mathcal{A}, k, n)$ も得られる．よって，$\forall n.\neg notAcc'(\mathcal{A}, k, n)$ である．

次に，$\forall n.\neg notAcc'(\mathcal{A}, k, n)$ ならば $\forall n.\neg notAcc(\mathcal{A}, k, n)$ であることを，その対偶が成立することを示すことで，明らかにする．ある n において $notAcc(\mathcal{A}, k, n)$ であると仮定する．このとき，定理 4.3.3 より，ある k-ループ σ が存在し，$\mathcal{A}, \sigma$ についての行程グラフ $G_\sigma^{\mathcal{A}}$ の初期ノードからの任意のパスにおいて受理ノードの出現が n 回以下である．ここで，そこからの任意のパスで受理ノードの出現が j 回以下である $G_\sigma^{\mathcal{A}}$ のノードの集合を V_j する．このとき，そのノード集合列 $V_0, \ldots, V_n$ は

$$I_\sigma^{\mathcal{A}}(V_0) \wedge \bigwedge_{0\le j<n} T_\sigma^{\mathcal{A}}(V_j, V_{j+1}) \wedge F_\sigma^{\mathcal{A}}(V_n) \wedge \bigwedge_{0\le j<n} V_j \subseteq V_{j+1}$$

を満たす．この $V_0, V_1, \ldots V_n$ から，以下を満たす $V'_0, V'_1, \ldots V'_{n'}$ を導ければよい．

$$\begin{aligned}&I_\sigma^{\mathcal{A}}(V'_0) \wedge \bigwedge_{0\le j<n'} T_\sigma^{\mathcal{A}}(V'_j, V'_{j+1}) \wedge F_\sigma^{\mathcal{A}}(V'_n) \wedge\\&\bigwedge_{0\le j<n'} V'_j \subseteq V'_{j+1} \wedge \bigwedge_{0\le j<n'} V'_j \neq V'_{j+1}\end{aligned}$$

このようなノード集合列は，$V_j = V_{j+1}$ のとき $V_0, \ldots, V_j, V_{j+2} \ldots V_n$ とするように，ノード集合列を変形していくことで得られる．

以上より，$\neg induction(\mathcal{A}, k, d)$ かつ $\neg notAcc(\mathcal{A}, k, d)$ ならば，任意の n でも $\neg notAcc(\mathcal{A}, k, n)$ であることが示される． □

条件 $induction(\mathcal{A}, k, d)$ の判定は，論理式 $|[induction(\mathcal{A}, k, d)]| :=$

$$|[loop(k)]| \wedge |[I(\mathcal{A}, k)]|_0 \wedge \bigwedge_{0 \le i < d+1} |[T(\mathcal{A}, k)]|_{j,j+1} \wedge$$
$$\bigwedge_{0 \le j < d+1} \bigwedge_{\substack{q \in Q, \\ 0 \le i \le k}} (v^j_{(q,i)} \to v^{j+1}_{(q,i)}) \quad \bigwedge_{0 \le j < d+1} \bigvee_{\substack{q \in Q, \\ 0 \le i \le k}} (\neg v^j_{(q,i)} \wedge v^{j+1}_{(q,i)})$$

の SAT 判定に帰着することが可能である．したがって，インダクションを用いたインクリメンタル k-全受理判定は，以下のようになる．

定義 4.3.3 (インダクションを用いたインクリメンタル k-全受理判定)**.**
入力：*NBA* $\mathcal{A}$*,* バウンド k*.*
出力：「k-全受理である」 *or*「k-全受理でない」

1. $d := 0$.
2. $notAcc(\mathcal{A}, k, d)$ であるか，つまり，$|[notAcc(\mathcal{A}, k, d)]|$ が充足可能であるかを調べる．
   - $notAcc(\mathcal{A}, k, d)$ であるならば,「k-全受理でない」と判定.
   - そうでないならば，次へ.
3. $induction(\mathcal{A}, k, d)$ であるか，つまり，$|[induction(\mathcal{A}, k, d)]|$ が充足可能性であるか調べる．
   - $induction(\mathcal{A}, k, d)$ でないならば,「k-全受理である」と判定.
   - そうでないならば，次へ.
4. $d := d + 1$ として，*2* へ.

4.3.4 他の行程グラフ構成を基にした帰着

4.3.2 節と 4.3.1 節で，NBA に受理されない k-ループを行程グラフを用いて特徴付け，それを基に NBA が k-全受理でないかの判定をサイズが $\mathcal{O}(k^2 \cdot |F| \cdot |\delta|)$ の命題論理式の SAT 判定に帰着できることを示した．本節では，行程グラフの構成に変更を加えることで，$\mathcal{O}(k \cdot |Q| \cdot |\delta|)$ の帰着も可能になることを述べる．この帰着法は，大きなバウンド k で検査を行う場合や，受理状態数が多い場合，4.3.1, 4.3.2 節で与えた帰着法よりも効率的である．

ここで考える行程グラフ（区別するため (k,l)-行程グラフと呼ぶ）は，各ノードでその直前の l（(k,l)-ループの番地）からそこまでの間に受理状態を通過したかを表す情報も保持するように 4.3.1 節で与えた行程グラフを拡張したものである．形式的には以下のように定義される．

定義 4.3.4 ((k,l)-行程グラフ)**.** *NBA* $\mathcal{A}$，(k,l)-ループ $\sigma = s_0 \ldots s_{l-1} (s_l \ldots s_k)^\omega$ に対して，(k,l)-行程グラフ $G = \langle V, v_I, E, C \rangle$ を，以下のように定義する．

- $V := Q \times \{0, 1, \ldots, k\} \times \{\top, \bot\}$.
- $v_I := (q_I, 0, \mathit{initb}(q_I))$．ここで，$\mathit{initb}(q_I)$ は，$q_I \in F$ とき $\top$，それ以外のとき $\bot$ を表す.
- $E := \{((q,i,b),(q',\mathit{suc}(i),\mathit{sucb}(i,b,q')) \mid (q,s_i,q') \in \delta\}$．ここで，$\mathit{sucb}(i,b,q')$ は，
  - $i < k$ のとき：$b = \top$ もしく $q' \in F$ のとき $\top$，それ以外のとき $\bot$.
  - $i = k$ とき：$q' \in F$ のとき $\top$，それ以外のとき $\bot$.
- $C := \{(q,k,\top) \mid q \in Q\}$

このように定義される (k,l)-行程グラフも，定理 4.3.1 と同様な性質を満たし，それゆえ，定理 4.3.4 と同様な性質も満たす．4.3.1 節の行程グラフとの違いは，その受理ノード数にある．4.3.1 節の行程グラフの受理ノード数が $(k+1) \cdot |F|$ であるのに対し，(k,l)-行程グラフの受理ノード数は $|Q|$ である（ノード数はともに $\mathcal{O}(k \cdot |Q|)$ である）．それゆえ，$\mathcal{O}(k \cdot |Q| \cdot |\delta|)$ の帰着が可能となる．

4.4 有界強充足可能性判定問題の計算量

本節では，バウンド k が 2 進数で表現されるとき（すなわち，k のサイズを $\lfloor \mathit{log}(k) \rfloor + 1$ とするとき），LTL で記述されたリアクティブシステム仕様の有界強充足可能性判定問題は co-NEXPTIME 完全であることを示す．すなわち，有界強充足可能性判定問題の補問題が，非決定性チューリング機械で $O(2^{p(n)})$ の時間を使って解ける問題のクラス NEXPTIME に属すること，及びそのクラス NEXPTIME に属する任意の問題が有界強充足可能性判定問題の補問題に多項式帰着可能であることを示す.

4.4.1 上界

まず，有界強充足可能性判定問題の計算量の上界，すなわちこの問題が co-NEXPTIME に属することを示す．ここでは，4.2 節の手続きを基に，有界強充足可能性判定の補問題が，非決定性チューリング機械で $O(2^{p(n)})$ の時間を使って解けることを示す．

定理 4.4.1. *LTL によるリアクティブシステム仕様と 2 進数で表現されたバウンドに対する有界強充足可能性問題は，co-NEXPTIME に属する．*

証明. 4.2 節の手続きのステップ 1 における $\mathcal{A}$ は，決定性チューリング機械においてでも，$\mathcal{O}(2^{|Spec|})$ の時間を用いて，$\mathcal{O}(2^{|Spec|})$ のサイズで構成可能である [90]．ステップ 2 の $\mathcal{A}'$ は，$\mathcal{A}$ の遷移関係に射影操作を施すことで得られるため，決定性チューリング機械においてでも，$\mathcal{O}(|\mathcal{A}|)$ の時間を使って，$\mathcal{O}(|\mathcal{A}|)$ のサイズで構成可能である．ステップ 3 の k-全受理の補問題は，定理 4.3.5 と定理 4.3.6 で示した通り，k(1 進数表現) と $|\mathcal{A}'|$ に対して多項式サイズの命題論理式の充足可能性判定問題に帰着される．よって，非決定性チューリング機械で $\mathcal{O}(p(|\mathcal{A}|+k))$ の時間を使って解くことができる．これより，有界強充足可能性問題の補問題は，非決定性チューリング機械で $\mathcal{O}(2^{p(|\mathcal{A}|+(\lfloor log(k) \rfloor+1))})$ の時間を使って解けることがわかる．ゆえに，有界強充足可能性判定問題は co-NEXPTIME に属する． □

4.4.2 下界

続いて，有界強充足可能性判定問題の計算量の下界，すなわち，この問題が co-EXPSPACE 困難であることを示す．ここでは，その計算量が NEXPTIME 完全であると知られている EXP-SQUARE TILING 問題（[21, 15] など）が有界強充足可能性判定問題の補問題へ帰着可能であることを示しことで，その co-NEXPTIME 困難性を明らかにする．

EXP-SQUARE TILING 問題とは，$\langle T, H, V, t_{init}, t_{final}, m \rangle$（ここで，$T$ は有限のタイル集合，$H, V \subseteq T \times T$ は横の隣接制約，縦の隣接制約であり，$t_{init}, t_{final} \in T$ は初期タイル，最終タイルである．m は 1 進数で表現された自然数である）に対して，以下の条件を満たすタイルの割り当て関数 $f : [0, \ldots, (2^m - 1)] \times [0, \ldots, (2^m - 1)] \to T$ が存在するかを決定する問題である．

- （初期タイルの条件）：$f(0, 0) = t_{init}$ である．

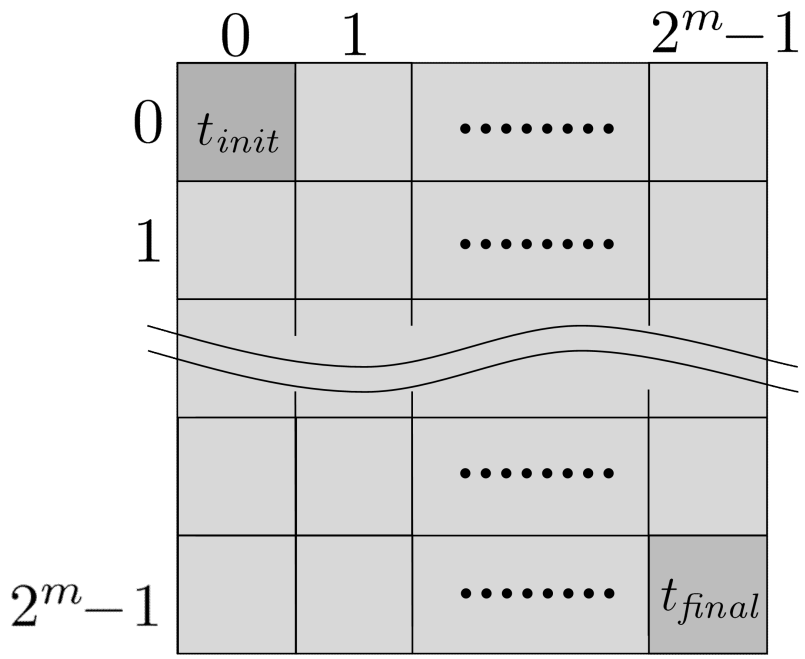

図 4.3: EXP-SQUARE TILING 問題

- （最終タイルの条件）: $f(2^m-1, 2^m-1) = t_{final}$ である.
- （横に隣接するタイルの条件）: 任意の $0 \leq i < 2^m-1, 0 \leq j \leq 2^m-1$ において，$(f(i,j), f(i+1,j)) \in H$ である.
- （縦に隣接するタイルの条件）: 任意の $0 \leq i \leq 2^m-1, 0 \leq j < 2^m-1$ において，$(f(i,j), f(i,j+1)) \in V$ である.

直感的には，図 4.3 に示すように，$2^m \times 2^m$ の格子の各マス目に対し，4 つの条件，つまり，初期タイル条件，最終タイル条件，横の隣接条件，縦の隣接条件を満たすようにタイルを置くことができるかを判定する問題である.（3.4 節の EXP-CORRIDOR TILING 問題では縦幅は非有界であったが，こちらの問題での縦幅は横幅と同じく 2^m である.）

上述のタイリング問題から有界強充足可能性判定問題の補問題への多項式帰着法，すなわち，タイリング問題 $\langle T, H, V, t_{init}, t_{final}, m\rangle$ に対して，その答えが「yes」であるとき，かつそのときにのみ，以下を満たすような φ_{tiling} と k_{tiling} の構成方法について述べる.

$$\exists \tilde{a}(\ \tilde{a} \text{ は } k_{tiling}\text{-ループである } \wedge \forall \tilde{b}(\langle \tilde{a}, \tilde{b}\rangle \models \neg\varphi_{tiling}))$$

ここで述べる帰着では，タイリング問題における「あるタイリングが存在する」を上式の「ある k_{tiling}-ループで表現される要求イベント集合列が存在する」に対応させる．そして,「そのタイリングが条件を満たす」を「そのタイリングに対応する要求イベント集合列については，任意の応答イベント集合列に対して LTL 式を満たすことがない」に対応させる．つまり，タイリング問題に対して，要求イベント集合列でその 1 つのタイリング（割り当て関数）を表現するように要求イベントを定める．そして，そのタイリングが条件を満たすことを，応答イベント集合列について全称束縛される式として表現するように応答イベント及び LTL 式を定める．

本帰着の基本的なアイディは 3.4 節の帰着と同様である．縦の隣接条件以外の表現においてはメタレベルの全称限量は必要なく，縦の隣接条件の表現においてメタレベルの全称限量を用いる．

要求イベント

要求イベントとしては，以下を用意する．

- x_t （$t \in T$）：格子のマス目 (i, j) においてタイル t が置かれるかを，時点 $i + (2^m) \cdot j$ において x_t が生起しているかに対応させる．
- end ：その時点でタイリングが終了しているかを表す．つまり，格子のマス目 (i, j) においてタイリングが終了しているかを時点 $i+(2^m)\cdot j$ において end が生起しているかに対応させる．
- $c_0, \ldots c_{2m-1}$ ：$2m$ ビットのカウンタとして機能させる．はじめはその値は 0 で，1 単位時間ごとにその値を 1 づつ増やす．このカウンターの上位 m ビット（下位 m ビット）は，その時点において着目しているマス目が何番目の行（列）であるかを表す．

応答イベント

応答イベントとしては，以下を用意する．

- $y_0, \ldots y_{m-1}$ ：縦の隣接制約を記述する際，着目する列を特定するのに用いる．

式： φ_{tiling}

式 φ_{tiling} を以下の式 (4.1)–(4.6) の論理積の否定とする．ただし，次を略記として用いる．

$$
\begin{aligned}
\overline{c} = 2^{2m} - 1 &\equiv \bigwedge_{0 \le i < 2m} \neg c_i \\
\overline{c}_{low} = 2^{m} - 1 &\equiv \bigwedge_{0 \le i < m} \neg c_i \\
\overline{c}_{high} = 2^{m} - 1 &\equiv \bigwedge_{m \le i < 2m} \neg c_i \\
\overline{c}_{low} = \overline{y} &\equiv \bigwedge_{0 \le i < m} (c_i \leftrightarrow y_i)
\end{aligned}
$$

- m ビットカウンター $c_0, \ldots c_{m-1}$ の制約.

$$
\left(\bigwedge_{0 \le i < m} \neg c_i \right) \quad \wedge
$$
$$
\mathbf{G} \left(\bigwedge_{0 \le i < m} \neg end \ \rightarrow \ \left((c_i \oplus \bigwedge_{0 \le j < i} c_j) \ \leftrightarrow \ \mathbf{X} c_i \right) \right) \tag{4.1}
$$

「はじめは $\overline{c}$ の指す値は 0 であり，タイリングが終了していなければ，1 単位時間ごとにその値が 1 づつ増える」ことを表す.

- マス目に張られるタイルを一つにする制約.

$$
\bigwedge_{t \in T} \mathbf{G} \left(x_t \ \rightarrow \ \bigwedge_{t' \neq t} \neg x_{t'} \right) \ \wedge \ \mathbf{G} \left(\neg end \ \rightarrow \ \bigvee_{t \in T} x_t \right) \tag{4.2}
$$

「各マス目に張られるタイルは高々一つ，かつ，そこでタイリングが終了していないならば，いずれかのタイルが張られる」ことを表す.

- 初期タイルの制約.

$$
x_{t_{init}} \tag{4.3}
$$

「マス目 $(0, 0)$ にタイル t_{init} が置かれる」ことを表す.

- 最終タイルの制約.

$$\neg end \mathbf{U}\Big(\neg end \wedge \overline{c} = 2^{2m} - 1 \wedge x_{t_{final}} \wedge \mathbf{XG} end\Big) \tag{4.4}$$

「マス目 $(2^m - 1, 2^m - 1)$（時点 $2^{2m} - 1$ に対応）に最終タイルが置かれ，タイリングが終了する」こと表す.

- 横の隣接制約.

$$\mathbf{G}\Big(\neg(\overline{c}_{low} = 2^m - 1) \wedge \neg end \rightarrow \bigvee_{(t,t') \in H} \big(x_t \wedge \mathbf{X} x_{t'}\big)\Big) \tag{4.5}$$

「現マス目が $2^m - 1$ の列でなく（右にマス目が存在する），かつタイリングが終了していないならば，現マス目に置くタイルと右マス目に置くタイルとが横の隣接制約を満たす」ことを表す.

- 縦の隣接制約.

$$\begin{aligned}
&\Big(\bigwedge_{0 \le i < m} \mathbf{G}(y_i \leftrightarrow \mathbf{X} y_i)\Big) \rightarrow \\
&\quad \mathbf{G}\Big(\Big(\overline{c}_{low} = \overline{y} \wedge \neg(\overline{c}_{high} = 2^m - 1) \wedge \neg end\Big) \rightarrow \\
&\qquad \bigvee_{(t,t') \in V} \Big(x_t \wedge \mathbf{X}\big((\overline{c} \neq \overline{y}) \mathbf{U} (\overline{c} = \overline{y} \wedge x_{t'})\big)\Big)\Big)
\end{aligned} \tag{4.6}$$

「$\overline{y}$ が不変であるならば，その $\overline{y}$ が示す列のマス目において，そのマス目が 2^m 番目の行でなく，タイリングが終了していないとき，現マス目と下のマス目とが縦の隣接制約を満たす」ことを表す．ここで,「現マス目と下のマス目とが縦の隣接制約を満たす」は,「現マス目に置くタイル t と，次に再び列が $\overline{y}$ となる最初のマス目，すなわち，同じ列にある直下のマス目に置くタイル t' とが $(t, t') \in V$ を満たす」として記述している.

これにより，$\forall \tilde{y}.\ (\ldots \models (4.6))$ が,「すべての列のおいて，縦の隣接制約を満たす」，つまり,「配置されるすべてのタイルが，縦の隣接制約を満たす」を表すことになる.

バウンド： k_{tiling}

バウンド k_{tiling} は 2^{2m} とする（この k_{tiling} のサイズは $2m+1$ である）．式 4.1 と 4.4 との制約により，$(2^{2m}, 2^{2m})$-ループのみが有界強充足可能性判定問題の反例となりうる．0 番目から $2^{2m}-1$ 番目の要素は，格子の各マス目に対応し，イベント *end* は，2^m 番目の要素にのみ含まれる．

定理 4.4.2. *LTL* によるリアクティブシステム仕様と *2* 進数で表現されたバウンド k に対する有界強充足可能性問題は，*co-NEXPTIME* 完全である．

証明. 上で示したように，EXP-SQUARE TILING 問題の答えが「yes」とき，かつそのときに限り，有界強充足可能とならない式 φ_{tiling} とバウンド k_{tiling} を構成することができる．φ_{tiling} はもとの問題に対して多項式のサイズであり，また，k_{tiling} のサイズは線形である．それらは多項式時間で構成可能であるため，EXP-SQUARE TILING 問題は有界強充足可能性判定の補問題に多項式帰着可能となる．EXP-SQUARE TILING 問題は NEXPTIME 完全であるため，有界強充足可能性判定の補問題は NEXPTIME 困難であり，有界強充足可能性判定問題は co-NEXPTIME 困難である．

□

4.5 実験

本章で与えた有界強充足可能性判定法を実装し，扱える仕様の規模・検査時間について他の性質判定との比較を行った．本節ではその結果を述べる．

4.5.1 実験内容

LTL で記述されたリアクティブシステム仕様を対象に，有界強充足可能性判定，非有界な強充足可能性判定，及び実現可能性判定を行い，各判定にかかる時間を計測・比較する．各性質判定は，以下のように行う．

有界強充足可能性判定には，次のような 4.2 節の手続きの実装を用いる．ステップ 1, 2 の NBA 構成（仕様を満たす応答が存在する要求イベント集合列を受理する NBA $\mathcal{A}'$ の構成）は，[9, 6] の実装法を基にする．ス

テップ 3 の $\mathcal{A}'$ の有界全受理判定には，定義 4.3.3 のインクリメンタルな判定法を採用する．NBA 構成や命題論理式生成などは，いずれも OCaml で実装し，SAT ソルバとしては MiniSat[28]（v.2.2）を用いる．

非有界な強充足可能性判定には，次のような 3.3 節の手続きの実装を用いる．ステップ 1, 2 の NBA 構成は，上の有界強充足可能性判定と同様であり，ステップ 3 の $\mathcal{A}'$ の全受理判定には，[96] によって提案された antichain ベースの手法を採用する．こちらも OCaml で実装したものを用いる．

実現可能性判定には，既存の実現可能性判定ツール：Acacia+[16](v.2.1) と Unbeast[30]（v.0.6b）を用いる（NBA 構成には LTL3BA[12](v.1.0.2) を用いる）．

本実験に用いるリアクティブシステム仕様は，以下の 5 種である．

- 付録に示す 2 種類の簡易 n 階建てエレベータの仕様 Ele_n と Ele_n^a,
- n プロセス-排他制御の仕様 Arb_n[69] と，その仕様の仮定部分の一部[1]を除去した Arb_n^{pa}.
- n クライアント-ロードバランサーの仕様 LB_n[30][2].

（Ele_n^a, Arb_n, LB_n には欠陥がなく，Ele_n, Arb_n^{pa} には欠陥がある.）

実験に利用する計算機環境は，CPU： Intel(R) Core(TM) i7-3820 3.60GHz，メモリ： 32GB である．

4.5.2 結果

表 4.2 と表 4.3 に，各種判定にかかった時間を示す.「T/O」は，1200 秒経過しても検査が終了しなかったことを表す．また,「T/O/A」は，オートマトン構成が 1200 秒以内に終了しなかったことを表す.

Ele_n^a, Arb_n, LB_n の判定では，いずれも「その性質を満たす」が出力された．表 4.2 から，本手法は，より大きな仕様に単純な欠陥がないこと，すなわち「単純な要求パターンに対しては，仕様を満たすように応答できる」ことを効率的に示せたことがわかる．

[1] Arb_n の仮定部分から $\bigwedge_i(\neg r_i \wedge \mathbf{G}((r_i \not\leftrightarrow g_i) \rightarrow (r_i \leftrightarrow \mathbf{X}r_i)))$ を除去したものを用いる．

[2] [30] では，様々なバリエーションが用意されているが，本実験では [16] のベンチマークでも利用されているフルバージョンのロードバランサー仕様を用いる．

表 4.2: 各性質判定にかかった時間（Ele_n^a, Arb_n, LB_n）

	有界強充足可能性			強充足可能性	実現可能性	
	$k=0$	$k=2$	$k=4$		Acacia+	Unbeast
Ele_2^a	0.02	0.12	0.18	0.91	0.82	0.30
Ele_3^a	0.03	0.26	4.27	201.67	201.58	T/O
Ele_4^a	0.15	5.27	245.30	T/O	T/O	
Ele_5^a	1.66	125.54	T/O			
Ele_6^a	27.98	T/O				
Ele_7^a	533.35					
Ele_8^a	T/O/A					
Arb_2	0.01	0.03	0.04	0.12	0.68	872.15
Arb_3	0.04	0.12	0.20	4.94	359.43	T/O
Arb_4	0.36	1.44	1.84	218.22	T/O	
Arb_5	4.73	14.47	24.25	T/O		
Arb_6	79.99	244.76	414.10			
Arb_7	T/O	T/O	T/O			
LB_2	0.02	0.02	0.03	0.04	0.03	0.13
LB_3	0.03	0.03	0.03	0.09	0.05	0.18
LB_4	0.03	0.04	0.04	0.41	0.31	0.96
LB_5	0.03	0.05	0.07	1.06	16.54	1.39
LB_6	0.04	0.11	0.15	4.02	T/O	21.26
LB_7	0.10	0.28	0.39	10.13		61.04
LB_8	0.34	1.10	1.31	36.20		96.96
LB_9	1.50	3.74	4.36	91.02		640.51
LB_{10}	6.09	19.31	18.59	333.22		T/O
LB_{11}	29.48	50.51	78.72	780.48		
LB_{12}	221.81	263.26	346.45	T/O		
LB_{13}	T/O/A	T/O/A	T/O/A			

表 4.3: 各性質判定にかかった時間（Ele_n, Arb_7^{pa}）

	有界強充足可能性		強充足可能性	実現可能性	
	$k=0$	$k=1$		Acacia+	Unbeast
Ele_2	0.01		0.01	6.47	0.02
Ele_3	0.02		0.11	540.97	T/O
Ele_4	0.09		2.03	T/O	
Ele_5	0.83		50.66		
Ele_6	13.08		T/O		
Ele_7	341.19				
Ele_8	T/O/A				
Arb_2^{pa}	0.02	0.03	0.05	0.12	T/O
Arb_3^{pa}	0.06	0.08	1.03	4.11	
Arb_4^{pa}	0.42	0.62	30.93	570.79	
Arb_5^{pa}	4.99	7.75	964.09	T/O	
Arb_6^{pa}	82.64	16.175	T/O		
Arb_7^{pa}	T/O	T/O			

Ele_n, Arb_n^{pa} の強充足可能性判定と実現可能性判定では，いずれも「その性質を満たさない」が出力された．Ele_n の有界強充足可能性判定では，$k=0$ で「その性質を満たさない」が出力され，反例が検出された．Arb_n の場合は，$k=0$ での判定では「その性質を満たす」が出力されたが，$k=1$ での判定で「その性質を満たさない」が出力され，反例が検出された．表 4.3 から，本手法は，より大きな仕様を取り扱え，小さな反例（仕様を満たすよう応答することのできない要求パターン）の存在を短時間で示せたことがわかる．有界強充足可能性は強充足可能性の必要条件であるため，本手法によって，強充足可能性判定ではタイムアウトしてしまった $Ele_n(5 < n \leq 7)$, Arb_6^{pa} が，強充足不能であることも示せたことになる．また，有界強充足可能性は実現可能性の必要条件でもあるため，有界強充足可能性の反例は，実現可能性の反例としてもみることができる．したがって，本手法は，$Ele_n(3 < n \leq 7)$, $Arb_n^{pa}(4 < n \leq 6)$ が実現不能であることも示めたことになる．

このような結果が得られたのは，単に考慮する要求パターンを制限し，高速な SAT ソルバを用いたためだけでなく，SAT 問題への効果的な帰着法を与えたためでもある．本研究では，受理状態の出現回数も d と制限

し，インクリメンタルに検査する手法を提案した．欠陥が存在する多くの場合，比較的小さな d で反例を見つられ，$Ele_n(2 \leq n \leq 7)$ の場合では，$d = 0$ の段階で反例を検出できた．

4.6 議論

有界モデル検査のための SAT エンコード法

有界モデル検査のため，LTL や very weak alternating Büchi automata（以下，VWABA），weak alternating Büchi automata（以下，WABA）の SAT エンコード法（パスがその式・オートマトンに満たされない・受理されないことを表す論理式の生成法）が提案されている [14, 56, 78, 39]．LTL や VWABA の表現力は本研究で用いた NBA の表現よりも低い．一方，WABA の表現力は NBA の表現力と等しい．実際，NBA は WABA に変換できるため，NBA の有界全受理判定は，[39] の WABA のエンコード法を用いても行える．しかし，本研究で与えたエンコード手法は，WABA を経由してエンコードするよりも効率が良い．WABA を経由してエンコードする場合，その命題論理式のサイズは $O(k \cdot |Q|^2 \cdot |\delta|)$ となるが，本研究では $O(k \cdot |Q| \cdot |\delta|)$ のエンコード法を与えている．

有界実現可能性

[34, 29] で与えられている実現可能性判定手続きでは，手続きを単純化するため，本研究と同様にオートマトンの受理条件を制限している．すなわち，受理条件として,「受理状態の出現回数が有限回である」の代わりに,「受理状態の出現回数が高々 d である」という条件を用いている．しかし [34, 29] では，証拠や反例のサイズは制限しない．

[76] では，有界実現可能性という性質が導入され，SMT ソルバを用いた判定手続きが与えられている．有界実現可能性とは,「状態数 k の状態遷移系として表現されるリアクティブシステムが存在し，そのすべての振る舞いが仕様を満たす」という性質である．有界実現可能性判定においては，サイズ k の状態遷移系が探索される．一方，本研究で与えた有界強充足可能性判定では，より単純な k-ループが探索される．本研究で提案した手法は，このようなシンプルな構造で表現される反例を探索す

ることで，より大きな仕様の単純な欠陥を効率的に検出することを可能にしている．

第5章　実現可能性判定に適した LTLサブセット

前章まででは,「検証する性質の制限」により，仕様の欠陥を効率的に検出する手法について取り扱った．本章では,「仕様の構文制限」により，実現可能性判定の煩雑さを回避することを考える．

仕様の実現可能性判定が，極めて煩雑で計算コストの高い処理を伴うのは，それが仕様を満たす振る舞いをちょうど受理する決定性 ω オートマトンを基に行われることによる．決定性 ω オートマトンは通常，次のように構成される：まず，仕様から無限長の語の上の非決定性 Büchi オートマトンを構成し，そして Safra の構成法 [75] を用い，それを決定化する．Safra の構成法は，有限長の語のオートマトンの決定化で用いられる部分集合構成法に比べて極めて煩雑である．部分集合構成法では，元のオートマトンの状態集合を決定化後の状態とするのに対し，Safra の構成法では，元のオートマトンの状態集合をノードとする木を決定化後の状態とする．そのため，その構成のための処理は煩雑となり，各種効率化を施しにくい．過去に実装も試みられているが，状態が十数個のオートマトンの決定化が限界である [40]．これまでに Safra の構成法を用いない実現可能性判定法も提案はされいているが [52]，扱える仕様の規模は依然としてごく限らたものとなっている．

本章では，実現可能性判定で必要となる決定性 ω オートマトン構成を簡単化する観点から，LTL の構文を制限した LTL サブセット $\mathrm{LTL}^{\mathrm{ep}}$ とその双対である $\mathrm{LTL}^{\mathrm{gp}}$ を，決定性 ω オートマトンの構成法とともに提案する．$\mathrm{LTL}^{\mathrm{gp}}$ 仕様からはある構造的特徴を持つ非決定性 ω オートマトンを構成できる．提案する構成法では，その構造的特徴を利用し，部分集合構成法を基にその非決定性 ω オートマトンを決定化することが可能である．$\mathrm{LTL}^{\mathrm{ep}}$ についても，$\mathrm{LTL}^{\mathrm{gp}}$ との双対性を利用することで，部分集合構成法を基に決定性 ω オートマトンを構成できる．

5.1 LTL サブセット $\mathrm{LTL}^{\mathrm{ep}}$ と $\mathrm{LTL}^{\mathrm{gp}}$

本節では，決定性 ω オートマトン構成を簡単化する観点から LTL の構文を制限した LTL サブセット $\mathrm{LTL}^{\mathrm{ep}}$ と $\mathrm{LTL}^{\mathrm{gp}}$ を与える．

$\mathrm{LTL}^{\mathrm{ep}}$ と $\mathrm{LTL}^{\mathrm{gp}}$ の定義

$\mathrm{LTL}^{\mathrm{ep}}$ と $\mathrm{LTL}^{\mathrm{gp}}$ は，以下のように定義される．

定義 5.1.1 ($\mathrm{LTL}^{\mathrm{ep}}$ と $\mathrm{LTL}^{\mathrm{gp}}$ の構文)**.** イベントの集合 P が与えられたとき，P 上の $\mathrm{LTL}^{\mathrm{ep}}$ の式は以下のように定義される．

$$f := b \mid f \wedge f \mid f \vee f \mid \mathbf{X}f \mid f\,\mathbf{U}b \mid f\,\mathbf{R}f$$

$\mathrm{LTL}^{\mathrm{gp}}$ の式は，以下のように定義される．

$$f := b \mid f \vee f \mid f \wedge f \mid \mathbf{X}f \mid f\,\mathbf{U}f \mid f\,\mathbf{R}b$$

ここで，b は，P 上のブール式（$\neg$，$\wedge$，$\vee$ より構成される式）とする．

$\mathrm{LTL}^{\mathrm{ep}}$ は,「いつか必ず満たされなければならない制約（イベンチャリティ）はブール式によって記述されなければならない」という制限を LTL の構文に課したサブセットである．式が否定標準形であると仮定すると，$f_1\mathbf{U}f_2$ における f_2 が将来必ず満たされなけければならない制約となる．このサブセットでは，その f_2 がブール式となるように構文を制限をしている．

また，$\mathrm{LTL}^{\mathrm{gp}}$ は，$\mathrm{LTL}^{\mathrm{ep}}$ と双対な関係にあり,「満たされ続けなければならない制約はブール式によって記述されなければならない」という制限を LTL 構文に課したサブセットである．$f_1\mathbf{R}f_2$ における f_2 が満たされ続けなければならない制約であり，このサブセットでは，この f_2 がブール式となるように構文を制限をしている．

$\mathrm{LTL}^{\mathrm{ep}}$ で記述可能な仕様

$\mathrm{LTL}^{\mathrm{ep}}$ では，以下のような仕様記述が可能である．

仕様 1 :「常に，開ボタンが押されたならばドアを開く.」

$$\mathbf{G}(Button_{open} \rightarrow open) \tag{5.1}$$
$$(\equiv \bot\mathbf{R}(button_{open} \rightarrow open))$$

仕様 2 :「イベント a は，無限にしばしば生起する.」

$$\mathbf{GF}a \tag{5.2}$$
$$(\equiv \bot\mathbf{R}(\top\mathbf{U}a))$$

仕様 3 :「リクエストされたならば，いつかサービスする.」

$$\mathbf{G}(req \to \mathbf{F}serv) \tag{5.3}$$
$$(\equiv \bot\mathbf{R}(\neg req \vee (\top\mathbf{U}serv)))$$

仕様 4 :「イベント a_1 が生起し，その後イベント a_2 が生起するならば，その間ずっとイベント b が生起し続けなければならない.」

$$\mathbf{G}((a_1 \wedge \neg a_2 \wedge \mathbf{F}a_2) \to (b\ \mathbf{U}a_2)) \tag{5.4}$$
$$(\equiv \bot\mathbf{R}(\neg a_1 \vee a_2 \vee (\bot\mathbf{R}\neg a_1) \vee (b\ \mathbf{U}a_2)))$$

以上のように，LTL^{ep} では仕様記述でよく用いられる LTL 式を記述可能である．[94] で与えられている「m 台リフトを持つ n 階建てエレベータシステム」の仕様も，LTL^{ep} の範囲に収まるものであった.

LTL^{ep} で記述不能な仕様

LTL^{ep} で記述できないのは，以下のような仕様である.

仕様 5 :「いつか必ずイベント a_1 が生起する．その a_1 の生起以後，必ず a_2 が生起する．さらに，その a_2 の生起以後，必ず a_3 が生起する.」

$$\mathbf{F}(a_1 \wedge (\mathbf{F}(a_2 \wedge \mathbf{F}a_3)) \tag{5.5}$$
$$(\equiv \top\mathbf{U}(a_1 \wedge (\top\mathbf{U}(a_2 \wedge (\top\mathbf{U}a_3)))))$$

仕様 6 :「ある時点以降，イベント a が生起し続ける.」

$$\mathbf{FG}a \tag{5.6}$$
$$(\equiv \top\mathbf{U}(\bot\mathbf{R}a))$$

LTL^{gp} で記述可能な仕様と記述不能な仕様

LTL^{gp} では，仕様 1, 5, 6 は記述できるが，仕様 2, 3, 4 は記述できない.

$\mathrm{LTL}^{\mathrm{ep}}$ と $\mathrm{LTL}^{\mathrm{gp}}$ の組み合わせ

$\mathrm{LTL}^{\mathrm{ep}}$ と $\mathrm{LTL}^{\mathrm{gp}}$ では，簡潔な手続きで決定性 ω オートマトンを構成できる（次節以降でその構成を与える）．決定性 ω オートマトンにおいては，和合成や積合成も簡潔に行うことができるため，$\mathrm{LTL}^{\mathrm{ep}}$ 式と $\mathrm{LTL}^{\mathrm{gp}}$ 式とをブール演算子で結合することで得られる仕様でも，$\mathrm{LTL}^{\mathrm{ep}}$ や $\mathrm{LTL}^{\mathrm{gp}}$ と同様，その決定性 ω オートマトンを簡潔な手続きで構成できる（その構成法は 5.5.3 節で述べる）．

仕様 7 :「システムは，ある時点以降停止し続ける．停止していないときにリクエストされたならば，いつかサービスする.」

$$\mathbf{FG}stop \wedge \mathbf{G}(\neg stop \rightarrow (req \rightarrow \mathbf{F}serv)) \tag{5.7}$$

上の仕様 7 は，$\mathrm{LTL}^{\mathrm{ep}}$ 式でも $\mathrm{LTL}^{\mathrm{gp}}$ 式でもないが，$\mathrm{LTL}^{\mathrm{gp}}$ 式と $\mathrm{LTL}^{\mathrm{ep}}$ 式の連言とみることができる．したがって，この仕様についても簡潔な手続きで決定性 ω オートマトンを構成できる．

また，リアクティブシステムの仕様記述で，しばしば用いられる

$$ASSUME \rightarrow GUARANRTEE$$

の形式（*ASSUME* には，想定する環境の要件を記述し，*GUARANRTEE* には，システムが満たすべき要件を記述する）の仕様の多くも対処可能である．

仕様 8 :「リクエスト 2 が来ないことが無限にしばしばあるならば，常に，リクエスト 1 が来たらいつかサービスし，リクエスト 2 が来ているときはサービスしない.」

$$\begin{aligned}&(\mathbf{GF}\neg req_2) \ \rightarrow \ \mathbf{G}((req_1 \rightarrow \mathbf{F}serv) \wedge (req_2 \rightarrow \neg serv)) \\ &(\equiv (\mathbf{FG}req_2) \ \vee \ (\mathbf{G}((req_1 \rightarrow \mathbf{F}serv) \wedge (req_2 \rightarrow \neg serv))))\end{aligned} \tag{5.8}$$

上の仕様 8 は，$\mathrm{LTL}^{\mathrm{ep}}$ 式でも $\mathrm{LTL}^{\mathrm{gp}}$ 式でもないが，*ASSUMPTION* 部と *GURANRTEE* 部が，ともに $\mathrm{LTL}^{\mathrm{ep}}$ 式である（$\mathrm{LTL}^{\mathrm{ep}}$ 式の否定式は $\mathrm{LTL}^{\mathrm{gp}}$ 式であるため，$\mathrm{LTL}^{\mathrm{gp}}$ 式と $\mathrm{LTL}^{\mathrm{ep}}$ 式の選言とみることができる）．そのため，このような仕様も，$\mathrm{LTL}^{\mathrm{ep}}$ や $\mathrm{LTL}^{\mathrm{gp}}$ と同様に，簡潔な手続きで決定性 ω オートマトンを構成できる．

本手法で扱えない仕様は，$\mathrm{LTL}^{\mathrm{ep}}$ でも $\mathrm{LTL}^{\mathrm{gp}}$ でもない時間部分式を持つような仕様である．例えば，

仕様 9 :「常に，ストップボタンが押されたならば，アラームを出し続けた後必ず停止し，その後停止し続ける.」

$$\mathbf{G}(buttom_{stop} \rightarrow alarm\mathbf{U}(\mathbf{G}stop)) \tag{5.9}$$

などが挙げられる.

5.2 ω オートマトン

前章まででは，ω オートマトンとして，非決定性 Büchi オートマトンのみを用いたが，本章では，他の ω オートマトンも用いる．ここでは，それらを導入する.

非決定性 ω オートマトンは，五つ組 $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, Acc \rangle$ である．ここで,

- Σ は有限のアルファベット,
- Q は有限の状態の集合,
- $q_I \in Q$ は初期状態,
- $\delta \subseteq Q \times \Sigma \times Q$ は遷移関係,
- Acc : 受理コンポーネントである.
  - 受理条件の種類によって，受理コンポーネントの定め方は異なる．本章では，Büchi の条件， co-Büchi の条件， generalized Büchi の条件，generalized co-Büchi の条件，及び generalized Street(1) の条件を用いる.

Büchi の条件を持つ ω オートマトンを Büchi オートマトンと呼ぶ．他の受理条件を持つ ω オートマトンも同様の呼び方をする.

Σ 上の無限長の語 α の行程は，以下を満たす状態の無限列 $\varrho \in Q^\omega$ である.

- $\varrho[0] = q_I$ であり，かつ
- すべての $i \geq 0$ において $(\varrho[i], \alpha[i], \varrho[i+1]) \in \delta$ である.

行程が以下の受理条件を満たすとき，その行程を受理行程と呼ぶ.

Büchi の条件： $Acc = F \subseteq Q$ であり，

$$inf(\varrho) \cap F \neq \emptyset.$$

co-Büchi の条件： $Acc = F \subseteq Q$ であり，

$$inf(\varrho) \cap F = \emptyset.$$

generalized Büchi の条件： $Acc = \mathcal{F} \subseteq 2^Q$ であり，

$$\forall F \in \mathcal{F}. inf(\varrho) \cap F \neq \emptyset.$$

generalized co-Büchi の条件： $Acc = \mathcal{F} \subseteq 2^Q$ であり，

$$\forall F \in \mathcal{F}. inf(\varrho) \cap F = \emptyset.$$

generalized Street(1) の条件：$Acc = (\mathcal{E}, \mathcal{F}) \in 2^Q \times 2^Q$ であり，

$$\forall E \in \mathcal{E}. inf(\varrho) \cap E \neq \emptyset \implies \forall F \in \mathcal{F}. inf(\varrho) \cap F \neq \emptyset.$$

ここで，$inf(\varrho)$ は ϱ に無限回出現する要素の集合を表し，$inf(\varrho) = \{q \in Q \mid \forall i \exists j > i. \varrho[j] = q\}$ である．語 α に対して，受理行程が存在するとき，$\mathcal{A}$ は α を受理するという．$\mathcal{A}$ が受理する無限長の語の集合を $\mathcal{A}$ の受理言語といい，$L(\mathcal{A})$ で表す．

オートマトンがすべての $q \in Q$,$a \in \Sigma$ に対して $(q, a, q') \in \delta$ である q' が高々 1 つしか存在しないとき，そのオートマトンは決定的であるといい，そのオートマトンを決定性 ω オートマトンと呼ぶ．

なお，以降では，非決定性（Non-deterministic）と決定性（Determinisitic）をそれぞれ “N”， “D” と記す．また， Büchi ，Co-Büchi, Generalized Büchi，Generalized Co-Büchi と generalized Street(1) とをそれぞれ “B”，“C”，“GB”，“GC”，“GS1” と記し，オートマトンを “A” と記す．例えば，非決定性 co-Büchi オートマトンは “NCA” と記す．

5.3 $\mathrm{LTL}^{\mathrm{gp}}$ の非決定性 ω オートマトンの特徴

5.3.1 LTL から非決定性 ω オートマトンの構成

まず本節では，一般的な LTL 式からその式を満たす振る舞いをちょうど受理する非決定性 ω オートマトンの構成法を与える．ここで与える構成法は，[6] 等に基づくものである．

なおここでは，LTL 式は否定標準形（否定演算子 $\neg$ が命題の直前にのみ出現する式）になっているものとする．任意の LTL 式は，以下の等価性を基に否定標準形へと変換できる．

$$
\begin{aligned}
\neg\neg\varphi &\equiv f, \\
\neg(f_1 \wedge f_2) &\equiv (\neg f_1) \vee (\neg f_2), \\
\neg(f_1 \vee f_2) &\equiv (\neg f_1) \wedge (\neg f_2), \\
\neg\mathbf{X}f &\equiv \mathbf{X}\neg f, \\
\neg(f_1\mathbf{U}f_2) &\equiv \neg f_1\mathbf{R}\neg f_2, \\
\neg(f_1\mathbf{R}f_2) &\equiv \neg f_1\mathbf{U}\neg f_2.
\end{aligned}
$$

ここで与える構成手続きでは，オートマトンの状態を入力式の部分式の集合とする．その部分式集合は，そこで満たされるべき式の集合を表す．入力式のみからなる集合を初期状態とし，式の分解と状態遷移を繰り返すことでオートマトンを構成する．式の分解では，状態に対応する式集合を以下の等価性に基づいて分解し，

$$
\begin{aligned}
f_1\mathbf{U}f_2 &\equiv f_2 \vee (f_1 \wedge \mathbf{X}(f_1\mathbf{U}f_2)) \\
f_1\mathbf{R}f_2 &\equiv f_2 \wedge (f_1 \vee \mathbf{X}(f_1\mathbf{R}f_2))
\end{aligned}
$$

次に遷移可能な状態（次に満たすべき式の集合）とその状態への遷移条件（その時点で満たすべき式の集合に対応）の対の集合を求める．例えば，$f_1\mathbf{U}f_2$ を制約として持つ状態であれば，f_2 がその時点で満たされる場合，制約のない状態に遷移できるようにし，f_1 がその時点で満たされる場合，$f_1\mathbf{U}f_2$ を制約として持つ状態に遷移できるようにする．分解手続きで得られた遷移可能な状態に対しても，さらにそこから遷移可能な状態を求めてゆく．Until 式 $f_1\mathbf{U}f_2$ が満たされるためには，いつか必ず f_2 が満たされなければならない．そのため，f_2 が成立せず，f_1 が成り立ち続けるような行程を不受理にする必要がある．この構成法では，f_2 が成り立つと，$f_1\mathbf{U}f_2$ を含まない状態へ遷移可能であるので，同じ Until 式 $f_1\mathbf{U}f_2$ が無限に出現し続けない行程のみを受理とする．

定義 5.3.1 (分解手続き)**.**
入力：式集合 S
出力：$(\emptyset$，式集合，式集合$)$ の集合

1. （初期化）$D := \{(S, \emptyset, \emptyset)\}$ とする．

2. （分解）任意の $(Proc_i, Cur_i, Next_i) \in D$，式 $f_{ij} \in Proc_i$ について，以下の操作 *(a)* から *(g)* のいずれかを，f_{ij} の形に応じて適用する．これを任意の要素の $Proc_i$ が空となるまで繰り返す．ここで，$Proc'_i := Proc_i - \{f_{ij}\}, Cur'_i := Cur_i \cup \{f_{ij}\}$ とする．

   (a) $f_{ij} = p$ *or* $\neg p$ のとき，$(Proc_i, Cur_i, Next_i)$ を以下で置き換える．
   $(Proc'_i, Cur'_i, Next_i)$.

   (b) $f_{ij} = f_1^{\mathbf{X}}$ のとき，$(Proc_i, Cur_i, Next_i)$ を以下で置き換える．
   $(Proc'_i \cup \{f_1\}, Cur'_i, Next_i)$.

   (c) $f_{ij} = f_1 \wedge f_2$ のとき，$(Proc_i, Cur_i, Next_i)$ を以下で置き換える．
   $(Proc'_i \cup \{f_1, f_2\}, Cur'_i, Next_i)$.

   (d) $f_{ij} = f_1 \vee f_2$ のとき，$(Proc_i, Cur_i, Next_i)$ を以下で置き換える．
   $(Proc'_i \cup \{f_1\}, Cur'_i, Next_i)$,
   $(Proc'_i \cup \{f_2\}, Cur'_i, Next_i)$.

   (e) $f_{ij} = \mathbf{X}f$ のとき，$(Proc_i, Cur_i, Next_i)$ を以下で置き換える．
   $(Proc'_i, Cur'_i, Next_i \cup \{f^{\mathbf{X}}\})$.

   (f) $f_{ij} = f_1 \mathbf{U} f_2$ のとき，$(Proc_i, Cur_i, Next_i)$ を以下で置き換える．
   $(Proc'_i \cup \{f_2\}, Cur'_i, Next_i)$,
   $(Proc'_i \cup \{f_1\}, Cur'_i, Next_i \cup \{f_{ij}\})$.

   (g) $f_{ij} = f_1 \mathbf{R} f_2$ のとき，$(Proc_i, Cur_i, Next_i)$ を以下で置き換える．
   $(Proc'_i \cup \{f_2, f_1\}, Cur'_i, Next_i)$,
   $(Proc'_i \cup \{f_1\}, Cur'_i, Next_i \cup \{f_{ij}\})$.

   $p, \neg p \in Cur_i$ のとき，その $(Proc_i, Cur_i, Next_i)$ を削除する．

   $f_1 \mathbf{U} f_2, f_2 \in Cur_i$，かつ $f_1 \mathbf{U} f_2 \in Next_i$ のとき，$Next_i$ から $f_1 \mathbf{U} f_2$ を除去する．

   $f_1 \mathbf{R} f_2, , f_1, f_2 \in Cur_i$，かつ $f_1 \mathbf{R} f_2 \in Next_i$ のとき，$Next_i$ から $f_1 \mathbf{R} f_2$ を除去する．

$Proc_i$ は分解前の式の集合，Cur_i はその時点で満たすべき式の集合，$Next_i$ は次の時点で満たすべき式の集合である．$f^{\mathbf{X}}$ の意味は，f と同様である．Until 式 $f_1 \mathbf{U} f_2$ の分解において，f_2 が満たされれずに繰り越される

$f_1\mathbf{U}f_2$ と $\mathbf{X}(f_1\mathbf{U}f_2)$ の分解の結果得られる $f_1\mathbf{U}f_2$ を区別するため，Next 式 $\mathbf{X}f$ の分解の結果得られる式を $f^{\mathbf{X}}$ にしている．

定義 5.3.2 (LTL から NGBA の構成)**.**
入力：*LTL* 式 ψ
出力：*NGBA* $\mathcal{N}_\psi = \langle 2^P, Q, q_I, \delta, \mathcal{F}\rangle$

1. （初期化）$Q := \{q_I\}$，$q_I := \{\psi\}$，$\delta := \emptyset$ とする．

2. （状態遷移）$q \in Q$ に対して，定義 *5.3.1* の分解手続きを行った結果を D とする．このとき，以下のように Q, δ を更新する．
   $Q := Q \cup \{N \mid (\emptyset, C, N) \in D\}$,
   $\delta := \delta \cup \{(q, s, N) \mid (\emptyset, C, N) \in D, p \in C \Rightarrow p \in s, \neg p \in C \Rightarrow p \notin s\}$.
   これを Q，δ が変化しなくなるまで繰り返す．

3. （受理コンポーネント）$\mathcal{F}$ を次のように定める．
   $\mathcal{F} := \{F_f \mid f \in sub(\psi)$，かつ，$f$ は $f_1\mathbf{U}f_2$ の形をした式$\}$.
   ここで，$F_f := \{q \mid f \notin q\}$.

式 ψ に対して上の手続きを適用して得られる NGBA $\mathcal{N}_\psi$ は，ψ を満たす振る舞いをちょうど受理する．

5.3.2 LTL^{gp} から得られる非決定性 ω オートマトンの特徴

LTL^{gp} 式においては,「常に満たすべき制約」はブール式でのみ記述される．このような式から，定義 5.3.2 の手続きで非決定性 ω オートマトンを構成すると，構成される非決定性 ω オートマトンには以下のような特徴が現れる．

- 受理行程には，ただ一つの状態 q_{loop} のみが無限回現れる．すなわち，受理行程は，最終的に自己ループを持つ状態 q_{loop} に至り，その状態のみを遷移し続ける．

これは，構文を制限したことで，受理行程が以下を満たすためである（上の q_{loop} は Release 式のみからなる）．

- あるポイント以降において，Release 式以外の式が出現しない．

- Relese 式については，あるポイント以降ずっと出現し続ける，もしくは，あるポイント以降出現しなくなる，のいずれかである．

行程の i 番目の状態 q_i に部分式 f が現れるのは，前状態 q_{i-1} に f が出現して，その f が現状態に繰り越される場合（f が Until 式，Release 式のとき），もしくは，前状態 q_{i-1} に含まれる他の部分式（f を真部分式として持つ式）を分解した結果，f が新たに生成される場合のいずれかである．常に満たすべき制約がブール式であると，各部分式 f は，いずれ他の部分式から生成されることはなくなり，受理行程においては，同じ Release 式が出現し続けるのみとなる．

定理 5.3.1. ψ を $\mathrm{LTL}^{\mathrm{gp}}$ 式とし，$\mathcal{N}_\psi$ を定義 *5.3.2* の手続きによって得られる *NGBA* とする．$\mathcal{N}_\psi$ の任意の行程 $\varrho = q_0 q_1 \ldots$ について，ϱ が受理行程であるならば，かつそのときに限り，ϱ は以下を満たす．

$$\exists i.\forall j \geq i.(q_j = q_i \ \wedge \ q_i \text{ は } \mathit{Release} \text{ 式のみからなる})$$

証明. [$\Leftarrow$] 行程 ϱ において，あるポイント以降ずっと Release 式のみが現れるならば，受理条件の定義より，行程 ϱ は受理行程である．

[$\Rightarrow$] ϱ が受理行程であるとき，ψ の任意の部分式 f において，以下の条件が満たされることを示す．

$$\begin{aligned} \exists i.((\forall j \geq i.f \in q_j) \ \vee \ (\forall j \geq i.f \notin q_j)) \quad & (f \text{ が Release 式のとき}) \\ \exists i.\forall j \geq i.f \notin q_j \quad & (\text{その他のとき}) \end{aligned}$$

まず，$f = \psi$（入力の式）において，条件を満たすことを示す．分解手続きの特徴より，q_{i+1} に式 f が現れるのは，q_i に f が出現して，その f が現状態に繰り越される場合（f が Until 式，Release 式のとき），もしくは，状態 q_i に含まれる他の部分式（f を真部分式として持つ式）を分解した結果，f が新たに生成される場合のいずれかである．$f = \psi$ のときは，f を真部分式として持つ式は存在しないので，f が Until 式，Release 式であり，無限に出現し続ける，もしくは，ある i 以降出現しないのいずれである．これより，f が Until 式以外のときについては，条件が満たされることがわかる．f が Until 式のときについては，ϱ が受理行程であることより，Until 式が無限に出現し続けることはなく，後者が成立するため，条件が満たされることがわかる．

次に，式 f を真部分式として持つ任意の式が条件を満たすと仮定して，f も条件を満たすことを示す．仮定より，以降の状態で f を真部分式として

持つのがRelease式のみであり，以後ずっとそれらRelease式が出現し続けるポイントi_rが存在する．各Release式$g = g_1\mathbf{R}g_2$は，$\forall j \geq i_r.g_1\mathbf{R}g_2 \in q_j$を満たすので，手続きの定義より，$i_r$以降分解手続きで$g_1$が処理されることはない（$i_r$以降の遷移に対応する$(\emptyset, Cur, Next)$のおいて，$g_1 \notin Cur$である）．また，$\psi$がLTL^{gp}であることから，$g_2$はブール式である．よって，式$f$を真部分式として持つ式から新たに$f$が生成されることはなく，$i_r$以降に$f$が現れるとすれば，以前より出現していた$f$が繰り越され，残り続ける場合のみである．したがって，$f = \psi$ときと同様の理由により，fは条件を満たす． □

LTL^{gp}式の非決定性ωオートマトンには，このような特徴が現れるため，generalized Büchiの受理条件でなく，Büchiの受理条件やco-Bčuhiの受理条件によってでも，受理行程を定めることができる．

定義 5.3.3 (LTL^{gp}のNBA（NCA）構成)**.**
入力：LTL^{gp}式ψ
出力：*NBA(NCA)* $\mathcal{N}_\psi = \langle 2^P, Q, q_I, \delta, F_b(F_c)\rangle$

1–2. Q, q_I, δを，定義*5.3.2*と同様に求める．

3' （受理コンポーネント）Fを次のように定める．
*Büchi*の条件：
$F_b := \{q \in Q \mid \forall f \in q.f$は$f_1\mathbf{R}f_2$の形をした式である$\}$
*co-Büchi*の条件：
$F_c := \{q \in Q \mid \exists f \in q.f$は$f_1\mathbf{R}f_2$の形をした式でない$\}$

定理 5.3.2. ψをLTL^{gp}式とし，$\mathcal{N}_\psi$を定義*5.3.3*の手続きによって得られる*NBA(NCA)*とする．このとき，$\mathcal{N}_\psi$はψを満たす振る舞いをちょうど受理する．

証明. 定理5.3.1より． □

定理 5.3.3. ψをLTL^{gp}式とし，$\mathcal{N}_\psi$を定義*5.3.3*の手続きによって得られる*NBA(NCA)*とする．このとき，$\mathcal{N}_\psi$の状態数は，高々$2^{|temp(\psi)|} + 1$である．ここで，$temp(\psi)$は，ψの部分時間式（*Next*式，*Until*式，及び*Release*式）の集合を表す．

証明. 初期状態を除く，すべての状態が$f^{\mathbf{X}}, f_1\mathbf{U}f_2, f_1\mathbf{R}f_2$のいずれかの形をした式の集合であることから． □

先に与えたオートマトンの特徴は，以下のように形式化される．

定理 5.3.4. ψ を $\mathrm{LTL}^{\mathrm{gp}}$ 式とし，$\mathcal{N}_\psi$ を定義 *5.3.3* の手続きによって得られる *NBA(NCA)* とする．このとき，$\mathcal{N}_\psi$ は以下（以降では特徴 1 と呼ぶ）を満たす．

- 任意の $q \in F_b$（$q \notin F_c$）において，q から q' へ遷移可能であり，かつ q' から q へ遷移可能であるような q' が存在するならば，$q = q'$ である．

証明. 定義 5.3.3 と定理 5.3.1 より． □

この構造的特徴は，

- 状態 $q \in F_b(q \notin F_c)$ を含む強連結成分は，その状態のみからなる

ことを表してる．これは，簡潔な決定化が知られている very weak（もしくは 1-weak）オートマトン [36] の構造的特徴「すべての強連結成分は，単一の状態のみからなる」よりも弱い特徴となっている．

5.4 構造的特徴を持つ NBA（NCA）の決定化

$\mathrm{LTL}^{\mathrm{gp}}$ から構成される NBA（NCA）は,「状態 $q \in F_b(q \notin F_c)$ を含む強連結成分は，その状態のみからなる」という特徴（特徴 1 ）を持つため，部分集合構成法を基に決定化できる．本節では，その構成法について述べる．

5.4.1 前処理

まず準備として，特徴 1 を持つ NBA（NCA）$\mathcal{N} = \langle \Sigma, Q, q_I, \delta, F_b(F_c) \rangle$ は，以下の特徴（以降では特徴 2 と呼ぶ）を持つように変換できることについて述べる．

- 任意の $q \in F_b(q \notin F_c), q' \in Q, a \in \Sigma$ において，$(q', a, q) \in \delta$ ならば $(q, a, q) \in \delta$ である．

特徴１を持つNBA（NCA）において，特徴２の条件を満たさない状態 $q \in F_b(q \notin F_c)$ が存在したとする．このとき，次のような変形を加えたNBA（NCA）は，特徴２を満たし，かつ，その受理言語はもとのオートマトンと同じである：q と同じ次状態を持つ新たな状態 q_{proxy} （$q_{proxy} \notin F_b(q_{proxy} \in F_c)$ とする）を導入し，q への遷移を q_{proxy} への遷移に置き換える．

この変形は，形式的には以下のように定義される．$\mathcal{N} = \langle \Sigma, Q, q_I, \delta, F_b(F_c) \rangle$ に対して，$\mathcal{N}' = \langle \Sigma, Q', q_I, \delta, F'_b(F'_c) \rangle$ を次のように定める．

$$
\begin{aligned}
Q' :=& Q \cup \{q_{proxy} \mid q \in F_b(q \notin F_c)\} \\
\delta' :=& \{(q, a, q') \mid (q, a, q') \in \delta, q' \notin F_b(q' \in F_c)\} \\
& \cup \{(q, a, q'_{proxy}) \mid (q, a, q') \in \delta, q' \in F_b(q \notin F_c)\} \\
& \cup \{(q_{proxy}, a, q' \mid (q, a, q') \in \delta, q \in F_b(q \notin F_c), q' \notin F_b(q' \in F_c)\} \\
& \cup \{(q_{proxy}, a, q'_{proxy} \mid (q, a, q') \in \delta, q \in F_b(q \notin F_c), q' \in F_b(q' \notin F_c)\} \\
F'_b :=& F_b \quad (F'_c = F_c \cup \{q_{proxy} \mid q \notin F_c\})
\end{aligned}
$$

このような変形を行うと，$q \in F_b(q \notin F_c)$ への遷移は，q_{proxy} からのみである．q_{proxy} からの遷移は q からの遷移と同様であるため，$\mathcal{N}'$ は特徴２を満たす．また，特徴１も満たしたままである．

また，次の理由から，受理言語も変わらない．

$\mathcal{N}$ が σ を受理する．

$\Longleftrightarrow$ $\mathcal{N}$ に以下を満たす σ の行程 $\varrho = q_0 q_1 \ldots$ が存在する．

$$\exists q^{loop} \in F_b(q^{loop} \notin F_c). \exists i. \forall j \geq i. q_j = q^{loop}$$

$\Longleftrightarrow$ $\mathcal{N}'$ に以下を満たす σ の行程 $\varrho = q_0 q_1 \ldots$ が存在する．

$$\exists q^{loop} \in F_b(q^{loop} \notin F_c). \exists i. (q_i = q^{loop}_{proxy} \wedge \forall j \geq i+1. q_j = q^{loop})$$

$\Longleftrightarrow$ $\mathcal{N}'$ が σ を受理する．

したがって，以下が成立する．

定理 5.4.1. $\mathcal{N} = \langle \Sigma, Q, q_I, \delta, F_b(F_c) \rangle$ を特徴１を持つ *NBA(NCA)* とする．このとき，$\mathcal{N}$ は，特徴１，特徴２および $L(\mathcal{N}) = L(\mathcal{N}')$ を満たす $\mathcal{N}'$ へ変形できる．

前節で定義した手続きにおいて出力されるNBA（NCA）は,（上の変化をしなくとも）特徴２を持つ．

定理 5.4.2. ψ を LTL^{gp} 式とし，$\mathcal{N}_\psi = \langle \Sigma, Q, q_I, \delta, F_b(F_c) \rangle$ を ψ から定義 *5.3.3* の手続きによって構成される *NBC*（*NCA*）とする．このとき，$\mathcal{N}_\psi$ は特徴 2 を持つ．

証明. 状態 $q \in F_b (q \notin F_c)$ への遷移 $(q', a, q) \in \delta$ が存在したとする．このとき，q' に対する分解手続きの結果にこの遷移に対応する $(\emptyset, C', q)$ が存在する．$q = \{f_1 \mathbf{R} g_1, f_2 \mathbf{R} g_2, \ldots, f_n \mathbf{R} g_n\}$ （g_i はブール式）とする．分解規則より，C' には各 i の $f_i \mathbf{R} g_i, g_i$ 及び g_i を成立させるための式（g_i を成立させるための式とは，例えば，$g_i = p_1 \vee (p_2 \wedge p_3)$ ならば，p_1 のみ，もしくは p_2 と q_3）が少なくとも含まれており（他の式より生成される式が含まれていることもある），原子命題とその否定が両方同時に入っていることはない．状態 $\{f_1 \mathbf{R} g_1, f_2 \mathbf{R} g_2, \ldots, f_n \mathbf{R} g_n\}$ の分解結果には，$C \subseteq C'$ であるような $(\emptyset, C, q)$ が存在する．なぜなら，分解手続きによって，まず，第 1 要素に g_i が，第 3 要素に $f_i \mathbf{R} g_i$ のみが入る三つ組が生じ，それより C が $f_i \mathbf{R} g_i, g_i$ 及び g_i を成立させるための式だけからなるような $(\emptyset, C, q)$ が最終的に生成されるためである．このような $(\emptyset, C, q)$ の存在より，遷移 (q, a, q) の存在が導かれる． □

5.4.2 決定化

前節で与えた特徴 1「状態 $q \in F_b$（$q \notin F_c$）を含む強連結成分は，その状態のみからなる」と，先に与えた特徴 2「状態 $q \in F_b$（$q \notin F_c$）へ a で遷移できるならば，状態 q から q 自身へ a で遷移できる」を持つ NBA（NCA）は，煩雑な Safra の構成法でなく，より単純な部分集合構成法（決定化前のオートマトンの状態集合を決定化後のオートマトンの状態とする構成法）を基に，決定性 ω オートマトンへ変換することが可能となる．以下でその手続きを与える．

定義 5.4.1 (特徴を持つ NBA(NCA) から DGCA の構成)**.**
入力：特徴 1 と特徴 2 を持つ *NBA*（*NCA*）$\mathcal{N} = \langle \Sigma, Q, q_I, \delta, F_b(F_c) \rangle$
出力：*DGCA* $\mathcal{D} = \langle \Sigma, Q', q'_I, \delta', \mathcal{F}' \rangle$

1. （初期化）$Q' := \{\{q_I\}\}$，$q'_0 := \{q_0\}$，$\delta' := \emptyset$ とする．

2. （状態遷移）任意の $q' \in Q'$，$a \in \Sigma$ に対して， Q'，δ' を次のように更新する．
   $Q' := Q' \cup \{q'_{succ}\}$.

$\delta' := \delta' \cup \{(q', a, q'_{succ})\}$.
ここで，q'_{succ} は以下のように定める．
$q'_{succ} = \{q_{succ} \in Q \mid q \in q', (q, a, q_{succ}) \in \delta\}$.
これを Q', δ' が変化しなくなるまで繰り返す.

3. （受理条件）受理条件は次のように定める．
$\mathcal{F}' = \{F'_q \mid q \in F_b (q \notin F_c)\}$,
$F'_q := \{q' \in Q' \mid q \notin q'\}$.

定理 5.4.3 (決定化の正当性)**.** $\mathcal{N}$ を特徴 1 と特徴 2 を持つ *NBA*（*NCA*）とする．このとき，定義 *5.4.1* の手続きによって $\mathcal{N}$ から得られる *DGCA* $\mathcal{D}$ は，$L(\mathcal{D}) = L(\mathcal{N})$ を満たす.

証明. $[\alpha \in L(\mathcal{N}) \Rightarrow \alpha \in L(\mathcal{D})]$ $\alpha \in L(\mathcal{N})$ と仮定すると，$inf(\varrho_n) \cap F_b \neq \emptyset$（$inf(\varrho_n) \cap F_c = \emptyset$）となる受理行程 ϱ_n が存在する．ここで，特徴 1 より，その行程には F_b に含まれる複数の状態（F_c に含まない複数の状態）が無限にしばしば現れることはなく，$|inf(\varrho_n)| = 1$ である. よって，ある状態 $q_{loop} \in F_b$ （$q_{loop} \in Q \setminus F_c$）が存在し，

$$\exists i. \forall j \geq i : \varrho_n[j] = q_{loop} \tag{a}$$

である．次に α に対する $\mathcal{D}$ の行程について考える．$\mathcal{D}$ の構成の定義より，α に対して唯一の行程 ϱ_d が存在して，$\varrho_d[i] = S_i$ とおくと，

$$S_i = \{\varrho[i] \in Q \mid \varrho \text{ は } \alpha \text{ に対する } \mathcal{N} \text{ の行程}\}$$

である．(a) より，あるポイント i_{loop} 以降，ずっと $q_{loop} \in S_j$ である．したがって，$\forall j \geq i_{loop}. S_j \notin F'_{q_{loop}}$ であり，$inf(\varrho_d) \cap F'_{q_{loop}} = \emptyset$ を満たす．よって，ϱ_d は $\mathcal{D}$ の受理行程であり，$\alpha \in L(\mathcal{D})$ である．ゆえに，$\alpha \in L(\mathcal{N}) \Rightarrow \alpha \in L(\mathcal{D})$ である．

$[\alpha \in L(\mathcal{D}) \Rightarrow \alpha \in L(\mathcal{N})]$ $\alpha \in L(\mathcal{D})$ と仮定すると，α に対する受理行程 $\varrho_d = S_0 S_1 \ldots$ が存在する．受理条件より，ある $q_{loop} \in F_b$（$q_{loop} \notin F_c$）が存在し，$inf(\varrho_d) \cap F'_{q_{loop}} = \emptyset$ である．つまり，ある i_{loop} 以降ずっと $\varrho_d[j] \notin F'_{q_{loop}}$ であり，

$$\forall j \geq i_{loop} : q_{loop} \in S_j \tag{b}$$

を満たす．ここで，$q_{loop} \in S_{i_{loop}}$ であるので，$\mathcal{N}$ において，

$$q_I \text{ から } q_{loop} \text{ へ } \alpha[0 \ldots i_{loop} - 1] \text{ で遷移可能} \tag{c}$$

である．また，(b) と特徴 2 より，

$$q_{loop} \text{ から } q_{loop} \text{ へ } \alpha[j] \text{ で遷移可能} \quad (j \geq i_{loop} \text{ において}) \tag{d}$$

である．これは，$\mathcal{D}$ の構成の定義より，$q_{loop} \in S_{j+1}$ であるならば，ある $q \in S_j$ が存在して，q から q_{loop} へ $\alpha[j]$ で遷移可能あり，特徴 2 より，q_{loop} からも q_{loop} へ $\alpha[j]$ で遷移可能であることがいえるためである．よって，(c) と (d) より，次の条件を満たす $\mathcal{N}$ の α に対する行程 ϱ_n が存在する．

$$\exists i.\forall j \geq i : \varrho_n[j] = q_{loop}$$

この行程は $inf(\varrho_n) \cap F_b \neq \emptyset$（$inf(\varrho_n) \cap F_c = \emptyset$）であることから，受理条件を満たす．よって，$\alpha \in L(\mathcal{N})$ である．ゆえに，$\alpha \in L(\mathcal{D}) \Rightarrow \alpha \in L(\mathcal{N})$ である． □

通常の NBA の決定化に用いられる Safra の構成法 [75] やその改良 [68] では，もとの NBA の状態集合をノードとする木（Safra's tree）を一つの状態とするオートマトンを構成する．その状態数のワーストケースは，NBA の状態数を n とすると，Safra の構成法では $(12)^n \cdot n^{2n}$，その改良では n^{2n+2} である [68]．それに対して，ここで与えた決定化手法では，もとの非決定性オートマトンの状態集合を一つの状態とするオートマトンを構成し，その状態数は高々2^n である．本決定化は，Safra の決定化に比べて簡潔であるため，BDD(Binary Decision Diagram) を利用するなど，実装段階において各種効率化を施しやすい．

定理 5.4.4. *$\mathcal{N}$ を特徴 1 と特徴 2 を持つ状態数 n の NBA（NCA）とする．このとき，定義 5.4.1 の手続きによって $\mathcal{N}$ から得られる DGCA $\mathcal{D}$ の状態数は高々2^n である．*

証明. $\mathcal{D}$ の状態集合 Q' が，$Q' \subseteq 2^Q$ であることによる． □

5.5 決定性 ω オートマトンの構成

本節では，決定性 ω オートマトン構成について述べる．

5.5.1 LTL^{gp} からの決定性 ω オートマトンの構成

LTL^{gp} 式からは，以下のように決定性 ω オートマトンを構成できる．

1. LTL^{gp} 式 ψ から構造的特徴を持つ NBA(NCA) $\mathcal{N}_\psi$ を定義 5.3.3 の手続きによって構成する．

2. 構造的特徴を持つ NBA(NCA) $\mathcal{N}_\psi$ を定義 5.4.1 の手続きによって決定化して，DGCA $\mathcal{D}_\psi$ を得る．

定理 5.5.1. ψ を LTL^{gp} 式とし，$\mathcal{D}_\psi$ を ψ から構成される *DGCA* とする．このとき，$\mathcal{D}_\psi$ の状態数は，高々$2^{2^{|temp(\psi)|+1}}$ である．

証明. 定理 5.3.3 と定理 5.4.4 より □

5.5.2 LTL^{ep} からの決定性 ω オートマトンの構成

LTL^{ep} 式についても，LTL^{gp} の DGCA 構成法を基に，その決定性 ω オートマトンを得ることができる．

LTL^{ep} と LTL^{gp} とは双対な関係にあり，LTL^{ep} では Until 式にのみ，LTL^{gp} では Release 式にのみ，それぞれ同様の制限を課している．それゆえ，以下が成立する．

定理 5.5.2. ψ を LTL^{ep} 式とする．このとき，$\neg\psi$（を否定標準形に変形した式）は LTL^{gp} 式である．

任意のオートマトンは全域的な（各状態ですべての $a \in \Sigma$ に対して遷移先が存在する）ものへと変換できる．全域的な決定性オートマトンおいては，受理条件を双対なものへと変更することで，その受理言語の補集合を認識する決定性オートマトンを得ることができる．

定理 5.5.3. $\mathcal{D}_{gc} = \langle \Sigma, Q, q_I, \delta, \mathcal{F}_{gc} \rangle$ を *DGCA* とし，$\mathcal{D}'_{gb} = \langle \Sigma, Q \cup \{q_*\}, q_I, \delta', \mathcal{F}' \rangle$ を *DGBA* とする．ここで，$T_* := \{(q_*, a, q_*) \mid a \in \Sigma\}$ とおき，

$\delta' := \delta \ \cup \ \{(q, a, q_*) \mid \neg\exists q'.(q, a, q') \in \delta\} \ \cup \ T_*$,

$\mathcal{F}' := \{F \cup \{q_*\} \mid F \in \mathcal{F}\}$.

とする．このとき，$L(\mathcal{D}'_{gb}) = \Sigma^\omega \setminus L(\mathcal{D}_{gc})$ である．

証明. DGCA $\mathcal{D}'_{gc} = \langle \Sigma, Q \cup \{q_*\}, q_I, \delta', \mathcal{F}' \rangle$ とすると，$L(\mathcal{D}'_{gc}) = L(\mathcal{D}_{gc})$ であり，また，$\mathcal{D}'_{gc}$ と $\mathcal{D}'_{gc}$ はすべての $\alpha \in \Sigma^\omega$ に対して必ずある同一の行程 ϱ が存在する．また，

$$
\begin{aligned}
\alpha \in L(\mathcal{D}'_{gb}) &\iff \forall F \in \mathcal{F}'.inf(\varrho) \cap F \neq \emptyset \\
&\iff \neg\exists F \in \mathcal{F}'.inf(\varrho) \cap F = \emptyset \\
&\iff \alpha \notin L(\mathcal{D}'_{gc})
\end{aligned}
$$

である．よって，$L(\mathcal{D}'_{gb}) = \Sigma^\omega \setminus L(\mathcal{D}'_{gc})$ であり，$L(\mathcal{D}'_{gb}) = \Sigma^\omega \setminus L(\mathcal{D}_{gc})$ である． □

したがって，LTL^{ep} 式 ψ からは，以下のように，ψ を満たす振る舞いをちょうど受理する決定性 ω オートマトンを構成できる．

1. LTL^{ep} 式 ψ から，LTL^{gp} 式 $\neg\psi$ （否定標準形になっているものとする）を得る．

2. LTL^{gp} 式 $\neg\psi$ から構造的特徴を持つ NBA(NCA) $\mathcal{N}_{\neg\psi}$ を定義 5.3.3 の手続きによって構成する．

3. 構造的特徴を持つ NBA(NCA) $\mathcal{N}_{\neg\psi}$ を定義 5.4.1 の手続きによって決定化して，DGCA $\mathcal{D}_{\neg\psi}$ を得る．

4. DGCA $\mathcal{D}_{\neg\psi}$ から, 定理 5.5.3 で与えた方法で，その補集合を認識する DGBA $\mathcal{D}_\psi$ を構成する．

定理 5.5.4. ψ を LTL^{ep} 式とし，$\mathcal{D}_\psi$ を ψ から構成される *DGBA* とする．このとき，$\mathcal{D}_\psi$ の状態数は，高々$2^{2^{|temp(\psi)|+1}} + 1$ である．

証明. 定理 5.5.1，$temp(\psi) = temp(\neg\psi)$ であること，及び DGCA から補集合を受理する DGBA への変換において状態数は 1 だけ増えることによる． □

5.5.3 LTL^{ep} と LTL^{gp} の組み合わせ仕様からの決定性 ω オートマトンの構成

LTL^{ep} 式・LTL^{gp} 式をブール演算子で結合することで得られる仕様からの決定性 ω オートマトン構成について考える．

決定性 ω オートマトンにおいては，比較的簡潔な手続きによって，和合成や積合成が可能である．したがって，LTL^{ep} 式・LTL^{gp} 式をブール演算子で結合することで得られる仕様についても，次のように決定性 ω オートマトンを簡潔に構成できる．

1. 各 $\mathrm{LTL}^{\mathrm{ep}}$ 部分・$\mathrm{LTL}^{\mathrm{gp}}$ 部分の決定性 ω オートマトンをそれぞれ構成する．

2. 各決定性 ω オートマトンを合成する．

5.1 節でも述べたように，*ASSUME* $\rightarrow$*GUARANRTEE* という形式が実際的な仕様記述ではしばしば用いられる．そこでここでは，($\mathrm{LTL}^{\mathrm{ep}}$ 式) $\rightarrow$ ($\mathrm{LTL}^{\mathrm{ep}}$ 式) という形式（($\mathrm{LTL}^{\mathrm{gp}}$ 式)$\vee$($\mathrm{LTL}^{\mathrm{ep}}$ 式) という形式）の仕様の決定性 ω オートマトン構成法を具体的に与える．

($\mathrm{LTL}^{\mathrm{ep}}$ 式)$\rightarrow$($\mathrm{LTL}^{\mathrm{ep}}$ 式) という形式の仕様の決定性 ω オートマトンは，仮定部 $\mathrm{LTL}^{\mathrm{ep}}$ 式と結論部 $\mathrm{LTL}^{\mathrm{ep}}$ 式の決定性 ω オートマトンをそれぞれ構成し，そられを合成することで得られる．ここでは，GS1 の受理条件を持つ決定性 ω オートマトンの構成法を与える．

ψ_a, ψ_g を $\mathrm{LTL}^{\mathrm{ep}}$ 式とする．$\mathrm{LTL}^{\mathrm{ep}}$ 式 ψ_a の DGBA を $\mathcal{D}^A = \langle \Sigma, Q^A, q_0^A, \delta^A, \mathcal{F}^{\mathcal{A}} \rangle$ とし，$\mathrm{LTL}^{\mathrm{ep}}$ 式 ψ_g の DGBA を $\mathcal{D}^G = \langle \Sigma, Q^G, q_0^G, \delta^G, \mathcal{F}^{\mathcal{G}} \rangle$ とする．ただし，両 DGBA は，定理 5.5.3 の DGBA のように全域的（$\forall q \forall a \exists q'.(q, a, q') \in \delta$)）になっていると仮定する．このとき，$\psi_a \rightarrow \psi_g$ を満たす振る舞いをちょうど受理する DGS1A $\mathcal{D} = \langle \Sigma, Q, q_0, \delta, (\mathcal{E}, \mathcal{F}) \rangle$ は，以下のように構成される．

- $Q := Q^A \times Q^G$.
- $q_0 := (q_0^A, q_0^G)$.
- $\delta := \{(q^A, q^G), a, (s^A, s^G))) \mid (q^A, a, s^A) \in \delta^A, (q^G, a, s^G) \in \delta^G\}$.
- $\mathcal{E} := \{A_F \mid F \in \mathcal{F}^{\mathcal{A}}\}$.
  ただし，$A_F := \{(q^A, q^G) \mid q^A \in F\}$.
- $\mathcal{F} := \{G_F \mid F \in \mathcal{F}^{\mathcal{G}}\}$.
  ただし，$G_F := \{(q^A, q^G) \mid q^G \in F\}$.

証明. $\sigma \in \Sigma^\omega$ とする．σ に対する $\mathcal{D}^A$ の行程を ϱ^A とし，σ に対する $\mathcal{D}^G$ の行程を ϱ^G とする（ともに全域的で決定的なオートマトンであるため，そのような行程が必ずひとつ存在する）．ϱ を σ に対する $\mathcal{D}$ の行程とする．$\mathcal{D}$ の δ の定義より，

$$\varrho[i] = (\varrho^A[i], \varrho^G[i]).$$

である．また，$\mathcal{E}, \mathcal{F}$ の定義より，任意の $F \in \mathcal{F}^{\mathcal{A}}$ において，

$$inf(\varrho^A) \cap F \neq \emptyset \Longleftrightarrow inf(\varrho) \cap A_F \neq \emptyset$$

であり，同様に，任意の $F \in \mathcal{F}^{\mathcal{G}}$ において，

$$inf(\varrho^G) \cap F \neq \emptyset \Longleftrightarrow inf(\varrho) \cap G_F \neq \emptyset$$

である．したがって，
$\sigma \in L(\mathcal{D}^A)$ ならば $\sigma \in L(\mathcal{D}^G)$.
$(\Longleftrightarrow)$ $\forall F \in \mathcal{F}^{\mathcal{A}}.inf(\varrho^A) \cap F \neq \emptyset$ ならば $\forall F \in \mathcal{F}^{\mathcal{G}}.inf(\varrho^G) \cap F \neq \emptyset$.
$(\Longleftrightarrow)$ $\forall A_F \in \mathcal{E}.inf(\varrho) \cap A_F \neq \emptyset$ ならば $\forall G_F \in \mathcal{F}.inf(\varrho) \cap G_F \neq \emptyset$.
$(\Longleftrightarrow)$ $\sigma \in L(\mathcal{D})$. □

定理 5.5.5. ψ_a, ψ_g を LTL^{ep} 式とする．このとき，$\psi_a \to \psi_g$ を満たす振る舞いをちょうど受理する *DGS1A* を高々$(2^{2^{|temp(\psi_a)|+1}}+1)\cdot(2^{2^{|temp(\psi_g)|+1}}+1)$ の状態数で構成可能である．

5.6 実現可能性判定

前節では，LTL^{ep}，LTL^{gp}，(LTL^{ep} 式)→(LTL^{ep} 式) 形式の仕様の決定性 ω オートマトン構成法を与えた．本節では，それを基にした実現可能性判定法について紹介する．紹介する判定法は，決定性 ω オートマトンを基に，実現可能性判定を無限ゲーム上の問題に帰着するものである．

5.6.1 無限ゲーム

本研究では，有向グラフ上の 2 プレイヤの無限ゲームを扱う．

ここで扱うゲームは，2 種の頂点（0-頂点と 1-頂点）を持つグラフの上で行われる．このゲームは直感的には以下のようなものである．ある頂点 v_0 にトークンが置かれている．その頂点 v_0 が，0-頂点のときはプレイヤ 0 が，1-頂点のときはプレイヤ 1 が，その頂点に隣接する次頂点 v_1 をひとつ選択し，そこにトークンを移動する．続いて v_1 でも同様に，トークンのある頂点 v_1 の種類に応じたプレイヤが隣接する次頂点 v_2 を選択し，トークンを移動する．これを繰り返す．もし，トークンが移動された頂点において隣接する次頂点が存在しない場合（その頂点が行き止まりであった場合）は，その頂点が 0-頂点であればプレイヤ 0 の負けとし，1-頂

点であればプレイヤ 1 の負けとする．行き止まりに陥らずにプレイが無限に続く場合は，勝利条件（ω オートマトンと受理条件と同様な方法で定められる）によって勝者が決定される．

このような無限ゲームの形式的な定義を以下に与える．ここで与える定義は，基本的に [60] に準じている．

無限ゲーム． 無限ゲーム $\mathcal{G} = \langle V_0, V_1, E, Win \rangle$ は，V_0:0-頂点の有限集合，V_1:1-頂点の有限集合，$E \subseteq (V_0 \cup V_1) \times (V_0 \cup V_1)$: 辺の集合，Win : 勝利条件からなる．$V_0 \cup V_1$ を V で表す．頂点 v の次頂点集合を $vE = \{v' \in V \mid (v, v') \in E\}$ と定義する．

プレイ． ゲーム $\mathcal{G}$ のプレイは，次のいずれかである．

- 無限プレイ：無限パス $\pi = v_0 v_1 \ldots \in V^\omega$ であり，任意の $i \in \mathbb{N}$ において $v_{i+1} \in v_i E$ である．
- 有限プレイ：有限パス $\pi = v_0 v_1 \ldots v_l \in V^+$ であり，任意の $i < l$ において $v_{i+1} \in v_i E$，かつ $v_l E = \emptyset$ である．

プレイ π が，以下のいずれかを満たすときプレイヤ 0 の勝利であるとする．

- 有限プレイであり，$\pi = \pi' v$ ，かつ $v \in V_1$ である．
- 無限プレイであり，勝利条件を満たす．

本論文では，勝利条件として，ω オートマトンの受理条件と同様のものを用いる．Büch の勝利条件を持つゲームを Büchi ゲームと呼ぶ．その他の勝利条件を持つゲームについても，同様の呼び方をする．

戦略． プレイヤ 0 の戦略は，部分関数 $f : V^* \times V_0 \to V$ であり，$f(\pi v)$ が定義されているときは必ず $f(\pi v) \in vE$ を満たす．プレイ $\pi = v_0 v_1 \ldots$ がすべての $v_i \in V_0$ において $v_{i+1} \in f(v_0 v_1 \ldots v_i)$ であるとき，プレイ $\pi = v_0 v_1 \ldots$ は f-適合するという．戦略 f は，v より始まるすべての f-適合するプレイがプレイヤ 0 の勝利であるとき，v において必勝であるという．v において必勝である戦略が存在するとき，プレイヤ 0 は v で勝利するといい，そのような頂点の集合 W_0 をプレイヤ 0 の勝利領域という．プレイヤ 1 についての戦略，必勝戦略や勝利領域も，プレイヤ 0 のときと同様に定義される．

各種勝利条件のゲームの必勝判定アルゴリズムは，それぞれ [60, 47] 等で与えられている．

5.6.2 無限ゲームを用いた実現可能性判定

リアクティブシステム仕様の実現可能性判定は，仕様の決定性 ω オートマトンを基に，次のように無限ゲーム上の問題に帰着できる．まず，決定性 ω オートマトンの遷移が要求による遷移と応答による遷移とからなると考え，要求による遷移を無限ゲームにおけるプレイヤ 1 のトークン移動に対応させ，応答による遷移をプレイヤ 0 のトークン移動に対応させる．すなわち，環境の振る舞いをプレイヤ 1 の振る舞いに，システムの振る舞いをプレイヤ 0 の振る舞いに対応させる．そして，振る舞いが決定性 ω オートマトンに受理される（振る舞いに対するオートマトンの行程が受理条件を満たす）ことを，ゲームにおけるプレイが勝利条件を満たすことに対応させる．このようなゲームでは，プレイヤ 0 の必勝戦略が仕様の実現と対応し，無限ゲームでプレイヤ 0 が必勝であるかが仕様が実現可能であるかと一致する．

以下に無限ゲームへの帰着による実現可能性判定法を与える．

1. 仕様 ψ より，前章で与えた方法で，$L(\mathcal{D}) = \{\sigma \in (2^{X \cup Y})^\omega \mid \sigma \models \psi\}$ である 決定性ω オートマトン $\mathcal{D} = \langle 2^{X \cup Y}, Q, q_0, \delta, Acc \rangle$ を構成する．

2. $\mathcal{D}$ より，ゲーム $\mathcal{G} = \langle V_0, V_1, E, Win \rangle$ を構成する．ここで，

   - $V_1 := Q$,
   - $V_0 := Q \times 2^X$,
   - $E := \{(q, (q, a)) \mid q \in Q, a \in 2^X\} \cup \{((q, a), q') \mid q \in Q, a \in 2^X, \exists b \in 2^Y.(q, a \cup b, q') \in \delta\}$
   - $Win = Acc$

   とする．

3. ゲーム $\mathcal{G}$ で，q_0 においてプレイヤ 0 が必勝であるかを調べる．必勝である場合,「実現可能である」と出力し，必勝でない場合,「実現不能である」と出力する．

5.7 実験

本章で与えた (LTL^{ep} 式)→(LTL^{ep} 式) 形式の仕様を対象とした決定性 ω オートマトン構成法，及び実現可能性判定法を実装し，扱える仕様の

規模，構成時間，判定時間について，構文の制限がない一般的な手法と比較を行った．

5.7.1 実験内容

提案した決定性 ω オートマトン構成法と実現可能性判定判定法の性能評価のため，以下のような実験を行った．

決定性 ω オートマトン構成法の性能評価．本章で与えた LTL サブセットで記述された仕様を対象に，提案した手法で決定性 ω オートマトンを構成し，構成される決定性 ω オートマトンの状態数・構成時間を測定する[1]．また，構文の制限がない一般的な LTL 仕様の決定性 ω オートマトン構成器 ltl2dstar[48] によっても，同じ仕様とその否定式の決定性 ω オートマトンを構成し[2]，状態数・構成時間を測定する．そして，それぞれを比較する．

実現可能性判定法の性能評価．本章で与えた LTL サブセットで記述された仕様を対象に，提案した手法によって実現可能性判定を行い[3]，その判定時間を測定する．また，構文の制限がない一般的な LTL 仕様の実現可能性判定器 Acacia+[16] と Unbeast[30] によっても同じ仕様の実現可能性判定を行い，その判定時間を測定する．そして，それぞれを比較する．

本実験に用いるリアクティブシステム仕様は，4.5 節の実験で用いた $Ele_n, Ele_n^a, Arb_n, Arb_n^{pa}, LB_n$ の 5 種である．Ele_n は，$\mathrm{LTL}^{\mathrm{ep}}$ 式であり，その他の仕様は，($\mathrm{LTL}^{\mathrm{ep}}$ 式)→($\mathrm{LTL}^{\mathrm{ep}}$ 式) の形式である．

提案手法の実装は，[6, 9] を基に OCaml で作成した．利用する計算機環境は，CPU： Intel(R) Core(TM) i7-3820 3.60GHz，メモリ： 32GB である．

5.7.2 結果

決定性 ω オートマトンの状態数と構成にかかった時間を表 5.1 と表 5.2 に示す．また，実現可能性判定にかかった時間を表 5.3 に示す.「T/O」は，

[1]状態数削減のため，[63] で提案されているテクニックを用いて，オートマトンを簡約する．

[2]否定式でも決定性 ω オートマトンを構成するのは，受理条件を変更することで，否定式の決定性 ω オートマトンからも元の仕様の決定性 ω オートマトンを得られるためである．

[3]こちらの実験では，生成された決定性 ω オートマトンに対して簡約を適用させない（ゲームの必勝判定よりも時間がかかることが多いため）．

1200 秒経過しても構成・検査が終了しなかったことを表す．Ele_n, Arb_n^{pa} の判定結果はすべて「実現不能である」であり，Ele_n^a, Arb_n, LB の判定結果はすべて「実現可能である」であった．

表 5.1 と表 5.2 より，本研究で提案した決定性 ω オートマトン構成法では，扱う仕様を限定しない一般的な LTL 仕様の決定性 ω オートマトン構成法に比べ，大きな仕様を扱え，より小さな決定性 ω オートマトンを高速に構成できたことがわかる．例えば，構文を制限しない手法においては，Ele_n では $n = 3$，Ele_n^a では $n = 2$ までしか決定性 ω オートマトンを構成できなかったのに対して，本手法では Ele_n では $n = 6$，Ele_n^a では $n = 5$ まで決定性 ω オートマトンを構成することができた．その構成時間も，本手法が大きく優っており，仕様を限定しない手法においては Ele_3 では 436.85 秒も構成にかかったのに対して，本手法では 0.02 秒とごく短時間で決定性 ω オートマトンを構成できた．また，得られるオートマトンの状態数も，本手法がすべての場合で優っていた．

表 5.3 より，本研究で提案した実現可能性判定法では，構文の制限がない一般的な実現可能性判定法に比べ，より大きな仕様を扱え，短時間で判定を行えたことがわかる．例えば，構文の制限がない手法においては，Ele_n, Ele_n^a ともに $n = 3$ までしか実現可能性の判定を行えなかったのに対して，本手法では，Ele_n では $n = 6$，Ele_n^a では $n = 5$ まで実現可能性の判定を行えた．その判定時間も，本手法が大きく優っており，仕様を限定しない手法においては Ele_3 の判定に 540.97 秒もかかったのに対して，本手法では 0.03 秒とごく短時間で実現可能性の判定を行えた．

以上のように，実現可能性を低コストに判定できる構文上の特徴を明らかにし，その特徴を満たす仕様の実現可能性を効率的に検査する，という本研究の目的が十分に果たされていることが確認された．

5.8 議論

他の構造的特徴を持つオートマトンとその LTL サブセット

構造的な特徴を持つオートマトンとして，weak オートマトンがよく知らている．weak オートマトンの条件を満たす NBA は，Persistent Property [59] と呼ばれる言語クラスと対応する（その補言語は Response Property [59] と対応する）[19]．また，対応する LTL サブセットも明らかにされている [20]．weak オートマトンは，「そのすべての強連結成分 $scc \subseteq Q$ が，

表 5.1: 状態数と構成時間（$Ele_n, Ele_n^a, Arb_n, Arb_n^{pa}$）

	本手法		ltl2dstar		ltl2dstar（否定の式）	
	状態数	構成時間	状態数	構成時間	状態数	構成時間
Ele_2	17	0.02	34	1.57	64	1.55
Ele_3	88	0.03	593	598.91	1895	436.85
Ele_4	599	0.14		T/O		T/O
Ele_5	4247	6.20				
Ele_6	29449	1102.96				
Ele_7		T/O				
Ele_2^a	35	0.02	68	27.20		T/O
Ele_3^a	219	0.05		T/O		
Ele_4^a	1853	1.62				
Ele_5^a	16523	394.23				
Ele_6^a		T/O				
Arb_2	31	0.02	176	0.87	3752	0.51
Arb_3	103	0.04	2567	93.23		T/O
Arb_4	351	0.26		T/O		
Arb_5	1247	6.18				
Arb_6	4607	250.34				
Arb_7		T/O				
Arb_2^{pa}	32	0.02	117	0.51	35138	5.27
Arb_3^{pa}	117	0.03	2854	72.00		T/O
Arb_4^{pa}	422	0.12		T/O		
Arb_5^{pa}	1491	4.36				
Arb_6^{pa}	5168	630.63				
Arb_7^{pa}		T/O				

表 5.2: 状態数と構成時間（LB_n）

	本手法		ltl2dstar		ltl2dstar（否定の式）	
	状態数	構成時間	状態数	構成時間	状態数	構成時間
LB_2	19	0.02	23	0.15	829	0.20
LB_3	28	0.02	36	0.66	19347	12.90
LB_4	36	0.02	48	7.46		T/O
LB_5	44	0.04	60	110.86		
LB_6	52	0.05		T/O		
LB_7	60	0.11				
LB_8	68	0.36				
LB_9	76	1.45				
LB_{10}	84	6.32				
LB_{11}	92	1160.88				
LB_{12}		T/O				

$scc \subseteq F$ であるか，$scc \cap F \neq \emptyset$ のいずれかである」という特徴を持つオートマトンのことである．この条件は，本研究で与えた特徴 1 の条件よりも緩いが，weak NBA は部分集合構成法を基に決定化できない．MH 構成法 [62] で決定化できることは知らているが [51]，この構成法は部分集合構成法よりも煩雑である[4]．

また，構造的特徴を持つオートマトンとして，terminal オートマトンと very weak オートマトン (1-weak オートマトンとも呼ばれる) もよく知られている．terminal オートマトンの特徴を持つ NBA は，Gurantee Property[59] と呼ばれる言語クラスと対応する．また，その補言語は Safety Property[59] と対応する [19]．それらの LTL サブセット $\mathrm{LTL}^{\mathrm{guarantee}}$ と $\mathrm{LTL}^{\mathrm{safety}}$ も与えられている [20]．very weak オートマトンの特徴を持つ NBA の補言語は，LTL と CTL の共通部分と対応することが知られおり，その LTL サブセット $\mathrm{LTL}^{\mathrm{det}}$ も与えられている [58].

terminal オートマトンや very weak オートマトンの条件を持つ NBA については，特徴 1 を持つ NBA と同様，部分集合構成法を基に決定化できる．[29] では，仕様を Safety Property と対応する部分とそうでない部分を分け，Safety Property と対応する部分については部分集合構成法を基

[4]部分集合構成法が，元のオートマトンの状態の集合を決定化後の状態にするのに対し，MH 構成法は，元のオートマトンの状態の集合のペアを決定化後の状態とする．

表 5.3: 実現可能性判定にかかった時間（秒）

	本手法	Acacia+	Unbeast
Ele_2	0.02	6.47	0.02
Ele_3	0.03	540.97	T/O
Ele_4	0.16	T/O	
Ele_5	3.98		
Ele_6	213.17		
Ele_7	T/O		
Ele_2^a	0.03	0.82	0.30
Ele_3^a	0.12	201.58	T/O
Ele_4^a	3.24	T/O	
Ele_5^a	111.66		
Ele_6^a	T/O		
Arb_2	0.03	0.68	872.15
Arb_3	0.05	359.43	T/O
Arb_4	0.18	T/O	
Arb_5	1.07		
Arb_6	7.97		
Arb_7	94.32		
Arb_8	T/O		
Arb_2^{pa}	0.03	0.12	T/O
Arb_3^{pa}	0.05	4.11	
Arb_4^{pa}	0.32	570.79	
Arb_5^{pa}	2.51	T/O	
Arb_6^{pa}	83.19		
Arb_7^{pa}	T/O		

	本手法	Acacia+	Unbeast
LB_2	0.01	0.03	0.13
LB_3	0.01	0.05	0.18
LB_4	0.02	0.31	0.96
LB_5	0.03	16.54	1.39
LB_6	0.09	T/O	21.26
LB_7	0.36		61.04
LB_8	1.49		96.96
LB_9	7.12		640.51
LB_{10}	29.89		T/O
LB_{11}	132.35		
LB_{12}	756.20		
LB_{13}	T/O		

に決定性 ω オートマトンを構成というアプローチで，実現可能性判定・システム合成を効率化している．[31] では，LTL と CTL の共通部分と対応する LTL 仕様に対する簡潔な実現可能性判定法・システム合成法を提案している．

しかし，terminal オートマトンや very weak オートマトンの条件は，特徴 1 の条件よりも強い．terminal オートマトンは,「状態 $q \in F_b$ は，他の状態へ遷移しない（自己ループのみ）」という条件を満たすオートマトンのことあるが，特徴 1 では，状態 $q \in F_b$ から他の強連結成分への遷移は存在してもよい．また，very weak オートマトンは,「すべての強連結成分は，単一の状態のみなる」という条件を満たすオートマトンのことであるが，特徴 1 では，F_b に含まれない状態のみからなる強連結成分については，単一の状態からなる必要はない．LTL サブセットについても，$\mathrm{LTL}^{\mathrm{guarantee}} \subseteq \mathrm{LTL}^{\mathrm{gp}}$，$\mathrm{LTL}^{\mathrm{safety}} \subseteq \mathrm{LTL}^{\mathrm{ep}}$，及び $\mathrm{LTL}^{\mathrm{det}} \subseteq \mathrm{LTL}^{\mathrm{ep}}$ が成立する．

このように，本章で与えた LTL サブセットやオートマトンの条件は，部分集合構成法を基に決定性 ω オートマトンを構成できることが既知の LTL サブセットやオートマトンの条件に比べて緩い．

他の LTL サブセット

[4] では，$\mathrm{LTL}(\mathbf{F}, \mathbf{X})$ や $\mathrm{LTL}(\mathbf{G}, \mathbf{F})$（$\mathrm{LTL}(op_1, \ldots, op_n)$ は，演算子 $op_1, \ldots, op_n$ からなる式のみを許す LTL サブセット）などについての決定性 ω オートマトン構成と実現可能性判定の計算量が与えられている．また [53] では，$\mathrm{LTL}(\mathbf{G}, \mathbf{F})$ の簡潔な決定性 ω オートマトン構成法について考察されている．これらでは，既存の LTL サブセットについての計算量や効率化について考察されているのに対し，本研究では，決定性 ω オートマトンを簡潔に構成できる新たな LTL サブセットを与えている．その点で，これらの研究と本研究は異なる．

[69] では，LTL サブセットのシンボリックな実現可能性判定法・システム合成法が提案されている（ツールの開発も行われている [46]）．そこでは，Generalized Reactivity(1) と呼ばれる以下のような形式の仕様を対象としている．

$$
\begin{aligned}
& b_i^e \wedge \bigwedge \mathbf{G} b_t^e \wedge \bigwedge \mathbf{GF} b_g^e \\
\rightarrow \quad & b_i^s \wedge \bigwedge \mathbf{G} b_t^s \wedge \bigwedge \mathbf{GF} b_g^s
\end{aligned}
$$

ここで，b_i^e, b_g^e, b_i^s と b_g^s はそれぞれブール式であり，また，b_t^e と b_t^s は，それぞれ原子命題と $\mathbf{X}v$（v は原子命題）とブール演算子からなる式である．そこで与えられた判定・合成法は効率的である一方，この形式では本研究で扱った (LTL^{ep} 式 $\rightarrow$ LTL^{ep} 式) よりも強く構文を制限している．

また，[69] では，$\bigwedge_i \varphi_i^b \rightarrow \bigwedge_j \psi_j^b$,（$\varphi_i^b$ と ψ_i^b は決定性 Büchi オートマトンで表現可能な式）という形式で記述された仕様の判定・合成は，上の形式の問題に帰着できることが述べられている．その帰着では，各 φ_i^b と ψ_j^b からそれと等価な決定性 Büchi オートマトンを構築した後，そのオートマトンを上の形式の仕様へ変換する．本研究で与えた効率的で簡潔な決定性 ω オートマトンの構成法は，この変換の効率化に寄与する．各 φ_i^b や ψ_j^b が LTL^{ep} 式であった場合，本章で与えた決定性 ω オートマトンの構成法を用いることで，より単純な問題に変換でき，効率的に判定・合成を行える．

有界強充足可能性判定との比較

本章では，構文制限によって扱える仕様の規模を広げることを試みたが，4 章では，検証する性質の制限について考え，強充足可能性の有界検査法を与えた．ここでは，その有界強充足可能性判定と本章で与えた実現可能性判定を比較する．

本章で与えた実現可能性判定と有界可能性判定のベンチマーク結果を表 5.4（欠陥あり仕様）と表 5.5（欠陥なし仕様）に示す．扱った仕様は，いずれも (LTL^{ep} 式 $\rightarrow$ LTL^{ep} 式) の形式，もしくは LTL^{ep} 式である．各判定時間は，4.5 節と 5.7 節と同じものである（欠陥あり仕様に対する有界強充足可能性の判定時間については，反例が検出された最小の k でのものを示している）．欠陥あり仕様の実現可能性判定では，いずれも「その性質を満たさない」が出力され，有界強充足可能性判定では，Ele_n の場合は $k \geq 0$ で，Arb^{pa} の場合は $k \geq 1$ で,「その性質を満たさない」が出力された．欠陥なし仕様の判定結果は，いずれも「その性質を満たす」である．

表 5.4 と表 5.5 からわかるように，扱える仕様の規模・判定時間については，一概に優劣はつけられない．仕様によっては,「構文制限」の効果の方が大きく現れることもあるし,「検証性質の制限」の効果の方が大きく現れることもある．

効率についての優劣はつけれないが，それぞれには次のような利点・欠

表 5.4: 本章の手法と有界強充足可能性の判定時間（欠陥あり仕様）

	本章の手法	有界強充足可能性判定
Ele_5	3.98	0.83 ($k = 0$)
Ele_6	213.17	13.08 ($k = 0$)
Ele_7	T/O	341.19 ($k = 0$)
Arb_5^{pa}	2.51	7.75 ($k = 1$)
Arb_6^{pa}	83.19	161.75 ($k = 1$)
Arb_7^{pa}	T/O	T/O

表 5.5: 本章の手法と有界強充足可能性の判定時間（欠陥なし仕様）

	本章の手法	有界強充足可能性判定		
		$k = 0$	$k = 2$	$k = 4$
Ele_5^a	111.66	1.66	125.54	T/O
Ele_6^a	T/O	27.98	T/O	
Ele_7^a		533.35		
Arb_5	1.07	5.11	14.47	24.25
Arb_6	7.97	84.26	244.76	414.10
Arb_7	94.32	T/O	T/O	T/O
LB_{10}	29.89	6.09	19.31	18.59
LB_{11}	132.35	29.38	50.51	78.72
LB_{12}	756.20	221.81	263.26	346.45
LB_{13}	T/O	T/O	T/O	T/O

点がある．有界強充足可能性判定では，構文への制限はなく，多様な仕様の検査を行うことができる．しかし，有界強充足可能性判定では，単純な欠陥が存在しないことのみしか示せない．それに対して本章の手法では，扱える仕様の構文は限定されているが，実現可能性の判定を行うことができ，実現可能であれば実現システムの合成も可能である．

本手法で取り扱うことのできない仕様について

5.1 節で挙げた仕様 9 : $\mathbf{G}(buttom_{stop} \rightarrow alarm\mathbf{U}(\mathbf{G}stop))$ やその否定式は，特徴 1 を持つ NBA や DGCA では表現できない．そのため，このような仕様をそのまま本手法で取り扱うことは本質的に難しい．

しかしこの仕様は，新たな原子命題を導入することで，仕様の意図を保ったまま，次のような $\mathrm{LTL}^{\mathrm{ep}}$ 式に書き換えることができる．

$$\mathbf{G}(buttom_{stop} \rightarrow alarm\mathbf{U}stop') \wedge \mathbf{G}(stop' \leftrightarrow \mathbf{G}stop)$$

この書き換えでは，新たに導入した原子命題で仕様の部分式を置き換え，別途追加した制約で，その原子命題とその部分式の真偽を一致させている．

このような書き換えは，多くの仕様に適用可能である．3.5 節で述べたように，このような書き換えを行っても，その仕様が強充足可能であるかどうかは変わらない．しかし，実現可能性の判定を行う場合，すべての仕様に対して適用可能な訳ではない．新たな原子命題で置き換える部分式の真偽が将来の要求イベントの生起に影響を受ける場合，その仕様が実現可能であるかどうかは必ずしも一致しない[5]．このような書き換えについての理論的な考察，すなわち，この書き換えを行っても実現可能であるかが変わらない仕様の構文的特徴を明らかにすることなどは，今後の課題である．

また，別の対策としては,「本手法の部分的適用」が挙げられる．仕様全体が $\mathrm{LTL}^{\mathrm{ep}}$ 式や $\mathrm{LTL}^{\mathrm{gp}}$ 式でなくとも，仕様の一部が $\mathrm{LTL}^{\mathrm{ep}}$ 式や $\mathrm{LTL}^{\mathrm{gp}}$ 式であることは多い．そのような場合，次のようなアプローチをとることで，仕様全体を一般的な手法で扱うよりも効率的に決定性 ω オートマトン構成・実現可能性判定を行うことができる．

1. $\mathrm{LTL}^{\mathrm{ep}}$ や $\mathrm{LTL}^{\mathrm{gp}}$ で記述されている部分の決定性 ω オートマトンを本章で提案した手法で構成する．

[5] 例えば，$(\mathbf{G}x)\mathbf{U}y$（x は要求イベント，y は応答イベント）は実現可能であるが，$y'\mathbf{U}y \wedge \mathbf{G}(y' \leftrightarrow \mathbf{G}x)$ は実現不能である．

2. その他の部分の決定性 ω オートマトンを一般的な手法で構成する．

3. 1. と 2. で構成した決定性 ω オートマトンを一つの決定性 ω オートマトンに合成する．

[29] では，仕様に Safety Property である部分が存在する場合，その部分についは部分集合構成法を基にオートマトンを決定化するという方法で，実現可能性判定の効率化に成功している．本章で与えた手法を基にすれば，仕様のより多くの部分に対して部分集合構成法が適用可能となるため，より効率的に実現可能性判定を行えることが予想される．

第6章　まとめ

本研究では，リアクティブシステム仕様の実現可能性に関する検証における計算量削減のため，次の２つアプローチをとった．

(i) 検証する性質の制限

(ii) 仕様の構文制限

(i) の「検証する性質の制限」では，実現可能性の必要条件である強充足可能性に着目し，まず，その計算量について考察した．その結果，(a) LTL で記述されたリアクティブシステム仕様の強充足可能性判定問題は EXPSPACE 完全であること，(b) 時間演算子のネストを 2 までと制限しても，強充足可能性判定問題の計算量は下がらず，EXPSPACE 完全のままであることが明らかになった．それが EXPSPACE 完全であることから，強充足可能性判定には，2EXPTIME である実現可能性判定と比較して計算量理論的なアドバンデージがあることがわかった．強充足可能性は実現可能性の必要条件ではあるが，多くの実際的な仕様が強充足可能ならば実現可能でもあるので，リアクティブシステム仕様の欠陥を効率的に検出するために，実現可能性判定の代わりに強充足可能性判定を用いることは有効であると考える．

さらに本研究では，強充足可能性をより制限した有界強充足可能性の概念を導入し，その判定手続きを提案した．有界強充足可能性は,「サイズ k の繰り返し構造として表現される任意の要求イベント集合列に対して，仕様を満たす応答イベント集合列が存在する」という性質であり，仕様を満たすように応答できない単純な要求パターンが存在しないことを表す．強充足不能な仕様は，比較的小さな繰り返し構造として表現される反例（仕様を満たす応答が存在しない要求パターン）を持つことが多く，この性質の判定で多くの欠陥を検出できる．提案した判定手続きは，仕様を満たすように応答可能な要求イベント集合列をちょうど受理する NBA を構成した後，その NBA の有界全受理判定を高速な SAT ソルバを用いて行うものである．有界強充足可能性問題の計算量についても考察

し，LTL で記述されたリアクティブシステム仕様の有界強充足可能性判定問題が co-NEXPTIME 完全であることを示した．これにより，有界強充足可能性判定には，強充足可能性判定と比較して計算量理論的なアドバンデージがあることがわかった．また実験により，本手法が強充足可能性や実現可能性の判定に比べて大きな仕様を取り扱え，小さな繰り返し構造として表現される反例の存在を効率的に調べられることを示し，その有効性を明らかにした．

(ii) の「仕様の構文制限」では，実現可能性判定において必要となる決定性 ω オートマトン構成を簡単化する観点から構文を制限した LTL のサブセット $\mathrm{LTL}^{\mathrm{gp}}$ と $\mathrm{LTL}^{\mathrm{ep}}$ を，決定性 ω オートマトン構成法とともに提案した．ここで提案した $\mathrm{LTL}^{\mathrm{gp}}$ は，その仕様からある構造的特徴を持つ非決定性 ω オートマトンを構成できるという性質を持つ．提案した構成法では，その構造的特徴を利用し，通常 ω オートマトンの決定化に用いられる Safra の構成法よりも簡潔な部分集合構成法によって決定性 ω オートマトンを構成できる．$\mathrm{LTL}^{\mathrm{ep}}$ についても，$\mathrm{LTL}^{\mathrm{gp}}$ との双対性を利用することで，部分集合構成法を基に決定性 ω オートマトンを構成できる．構文を制限した仕様について実験を行い，本手法によってより大規模な仕様を取り扱え，効率的に決定性 ω オートマトン構成・実現可能性判定が行えることを示し，その有効性を明らかにした．

リアクティブシステム仕様の実現可能性判定のような難しい問題を取り扱うための現実的なアプローチは，入力や調べる性質を限定して効率的な解法を模索することである．一般的な実現可能性判定では，付録の簡易 n 階建てエレベータの仕様において，$n = 3$ まででしか検証を行えなかったが，(i) の「検証性質の制限」（有界強充足判定法）によって $n = 7$ までの仕様の欠陥を見つけられるようになった．また，(ii) の「仕様の構文制限」によって $n = 6$ までの仕様の判定を行えるようになった．実際的なシステムの仕様を直接取り扱えるほどとはいえないが，システムの重要箇所を抜き出した部分仕様については有効な検証を行える規模であるといえる．

今後の課題としては，(i) の「検証性質の制限」と (ii) の「仕様の構文制限」の組み合わせについて検討することが挙げられる．5 章で与えた $\mathrm{LTL}^{\mathrm{gp}}$ 仕様からは，構造的特徴を持つ NBA を構成することができる．3 章で扱った強充足可能性や 4 章で与えた有界強充足可能性の判定は NBA を基にするため，$\mathrm{LTL}^{\mathrm{gp}}$ 仕様の強充足可能性判定や有界強充足可能性判定は，より簡潔に遂行され得る．しかし，$\mathrm{LTL}^{\mathrm{gp}}$ 単体では実際的な仕様

を記述しづらい．より実際的で効率的な検証を行える「検証性質の制限」と「仕様の構文制限」の組み合わせ方法について今後検討していきたいと考えている．

謝辞

本研究を進めるにあたり，ご多忙の中ご指導を頂きました米崎直樹教授に心から感謝申し上げます．また，多くの助言とご支援を頂いた萩原茂樹助教，ならびに米崎研究室とその OB の皆様に感謝いたします．

参考文献

[1] Martín Abadi, Leslie Lamport, and Pierre Wolper. Realizable and unrealizable specifications of reactive systems. In *Proc. 16th International Colloquium on Automata, Languages, and Programming*, Vol. 372 of *LNCS*, pp. 1–17. Springer, 1989.

[2] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, Vol. 126, No. 2, pp. 183–235, 1994.

[3] Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *J. ACM*, Vol. 41, No. 1, pp. 181–203, 1994.

[4] Rajeev Alur and Salvatore La Torre. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Logic*, Vol. 5, No. 1, pp. 1–25, 2004.

[5] 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, pp. 262–281, 2011.

[6] Takenobu Aoshima, Kenji Sakuma, and Naoki Yonezaki. An efficient verification procedure supporting evolution of reactive system specifications. In *Proc. International Workshop on Principles of Software Evolution (IWPSE2001)*, pp. 182–185, 2001.

[7] Takenobu Aoshima and Naoki Yonezaki. Verification of reactive system specification with outer event conditional formula. In *Proc. International Symposium on Principles of Software Evolution (ISPSE2000)*, pp. 195–199, 2000.

[8] 青島武伸. リアクティブシステム仕様の検証系に関する研究. 博士論文, 東京工業大学, 2003.

[9] 青島武伸, 米崎直樹. 時間論理によるリアクティブシステム仕様の検証の効率化. コンピュータソフトウェア, Vol. 20, No. 3, pp. 30–53, 2003.

[10] Roy Armoni, Limor Fix, Ranan Fraer, Scott Huddleston, Nir Piterman, and Moshe Y. Vardi. SAT-based induction for temporal safety properties. *Electr. Notes Theor. Comput. Sci.*, Vol. 119, No. 2, pp. 3–16, 2005.

[11] Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*, pp. 469–474. Elsevier, 1998.

[12] Tomáš Babiak, Mojmír Křetínský, Vojtěch Řehák, and Jan Strejček. LTL to Büchi automata translation: Fast and more deterministic. In *Proc. 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Vol. 7214 of *LNCS*, pp. 95–109. Springer, 2012.

[13] Armin Biere. PicoSAT essentials. *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, Vol. 4, pp. 75–97, 2008.

[14] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In *Proc. 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, Vol. 1579 of *LNCS*, pp. 193–207. Springer, 1999.

[15] Peter Van Emde Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, Vol. 187 of *Lecture Notes in Pure and Applied Mathematics*, pp. 331–363. Marcel Dekker Inc, 1997.

[16] Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. Acacia+, a tool for LTL synthesis. In *Proc. 24th International Conference on Computer Aided Verification*, Vol. 7358 of *LNCS*, pp. 652–657. Springer, 2012.

[17] Patricia Bouyer, Deepak D'Souza, P. Madhusudan, and Antoine Petit. Timed control with partial observability. In *Proc. 15th International Conference on Computer Aided Verification*, Vol. 2725 of *LNCS*, pp. 180–192. Springer, 2003.

[18] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: A model-checking tool for real-time systems. In *Proc. 5th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, Vol. 1486 of *LNCS*, pp. 298–302. Springer, 1998.

[19] Ivana Cerna and Radek Pelanek. Relating hierarchy of temporal properties to model checking. In *Proc. 28th International Symposium on Mathematical Foundations of Computer Science*, Vol. 2747 of *LNCS*, pp. 318–327. Springer, 2003.

[20] Edward Y. Chang, Zohar Manna, and Amir Pnueli. Characterization of temporal property classes. In *Proc. 19th International Colloquium on Automata, Languages and Programming*, Vol. 4623 of *LNCS*, pp. 474–486. Springer, 1992.

[21] Bogdan S. Chlebus. From domino tilings to a new model of computation. In *Symposium on Computation Theory*, pp. 24–33, 1984.

[22] Alessandro Cimatti, Edmund M. Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. NuSMV 2: An opensource tool for symbolic model checking. In *Proc. 14th International Conference on Computer Aided Verification*, Vol. 2404 of *LNCS*, pp. 359–364. Springer, 2002.

[23] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proc. IBM Workshop on Logic of Programs*, Vol. 131 of *LNCS*, pp. 52–71. Springer, 1982.

[24] Stéphane Demri, Ranko Lazić, and David Nowak. On the freeze quantifier in constraint LTL: Decidability and complexity. *Information and Computation*, Vol. 205, No. 1, pp. 2–24, 2007.

[25] Stéphane Demri and Philippe Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Information and Computation*, Vol. 174, No. 1, pp. 84–103, 2002.

[26] Deepak D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *Proc. 19th Annual Symposium on Theoretical Aspects of Computer Science*, Vol. 2289 of *LNCS*, pp. 571–582. Springer, 2002.

[27] Bruno Dutertre and Leonardo de Moura. A fast linear-arithmetic solver for DPLL(T). In *Proc. 18th International Conference on Computer Aided Verification*, Vol. 4144 of *LNCS*, pp. 81–94. Springer, 2006.

[28] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Proc. 6th International Conference on Theory and Applications of Satisfiability Testing*, Vol. 2919 of *LNCS*, pp. 502–518. Springer, 2003.

[29] Rüdiger Ehlers. Symbolic bounded synthesis. In *Proc. 22nd International Conference on Computer Aided Verification*, Vol. 6174 of *LNCS*, pp. 365–379. Springer, 2010.

[30] Rüdiger Ehlers. Unbeast: Symbolic bounded synthesis. In *Proc. 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Vol. 6605 of *LNCS*, pp. 272–275. Springer, 2011.

[31] Rüdiger Ehlers. ACTL $\cap$ LTL synthesis. In *Proc. 24th International Conference on Computer Aided Verification*, Vol. 7358 of *LNCS*, pp. 39–54. Springer, 2012.

[32] Jacob Elgaard, Nils Klarlund, and Anders Møller. MONA 1.X: New techniques for WS1S and WS2S. In *Proc. 10th International Conference on Computer Aided Verification*, Vol. 1427 of *LNCS*, pp. 516–520. Springer, 1998.

[33] E. Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. In *Proc. 14th Annual ACM Symposium on Theory of Computing (STOC '82)*, pp. 169–180. ACM, 1982.

[34] Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. An antichain algorithm for LTL realizability. In *Proc. 21st International*

Conference on Computer Aided Verification, Vol. 5643 of *LNCS*, pp. 263–277. Springer, 2009.

[35] Emmanuel Filiot, Nayiong Jin, and Jean-François Raskin. Compositional algorithms for LTL synthesis. In *Proc. 8th International Conference on Automated Technology for Verification and Analysis*, Vol. 6252 of *LNCS*, pp. 112–127. Springer, 2010.

[36] Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In *Proc. 13th International Conference on Computer Aided Verification*, Vol. 2102 of *LNCS*, pp. 53–65. Springer, 2001.

[37] Shigeki Hagihara, Yusuke Kitamura, Masaya Shimakawa, and Naoki Yonezaki. Extracting environmental constraints to make reactive system specifications realizable. In *Proc. 16th Asia-Pacific Software Engineering Conference*, pp. 61–68. IEEE Computer Society, 2009.

[38] Shigeki Hagihara and Naoki Yonezaki. Completeness of verification methods for approaching to realizable reactive specifications. In *Proc. 1st Asian Working Conference on Verified Software, AWCVS'06*, Vol. 348 of *UNU-IIST*, pp. 242–257, 2006.

[39] Keijo Heljanko, Tommi A. Junttila, Misa Keinänen, Martin Lange, and Timo Latvala. Bounded model checking for weak alternating Büchi automata. In *Proc. 18th International Conference on Computer Aided Verification*, Vol. 4144 of *LNCS*, pp. 95–108. Springer, 2006.

[40] Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *Proc. 20th International Conference on Computer Science Logic*, Vol. 4207 of *LNCS*, pp. 395–410. Springer, 2006.

[41] C. A. R. Hoare. *Communicating sequential processes.* Prentice-Hall, 1985.

[42] Gerard J. Holzmann. The model checker SPIN. *IEEE Trans. Softw. Eng.*, Vol. 23, No. 5, pp. 279–295, 1997.

[43] Daniel Jackson. Automating first-order relational logic. In *Proc. 8th ACM SIGSOFT international symposium on Foundations of software engineering: twenty-first century applications*, pp. 130–139. ACM, 2000.

[44] Daniel Jackson. *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, 2006.

[45] Barbara Jobstmann and Roderick Bloem. Optimizations for LTL synthesis. In *Proc. 6th International Conference on Formal Methods in Computer-Aided Design*, pp. 117–124, 2006.

[46] Barbara Jobstmann, Stefan Galler, Martin Weiglhofer, and Roderick Bloem. Anzu: A tool for property synthesis. In *Proc. 19th International Conference on Computer Aided Verification*, Vol. 4590 of *LNCS*, pp. 258–262. Springer, 2007.

[47] Yonit Kesten, Nir Piterman, and Amir Pnueli. Bridging the gap between fair simulation and trace inclusion. In *Proc. 15th International Conference on Computer Aided Verification*, Vol. 2725 of *LNCS*, pp. 381–393. Springer, 2003.

[48] Joachim Klein and Christel Baier. Experiments with deterministic ω-automata for formulas of linear temporal logic. *Theor. Comput. Sci.*, Vol. 363, No. 2, pp. 182–195, 2006.

[49] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. Safraless compositional synthesis. In *Proc. 18th International Conference on Computer Aided Verification*, Vol. 4144 of *LNCS*, pp. 31–44. Springer, 2006.

[50] Orna Kupferman and Moshe Y. Vardi. Synthesizing distributed systems. In *Proc. 16th Annual IEEE Symposium on Logic in Computer Science*, pp. 389–398. IEEE Computer Society, 2001.

[51] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Comput. Logic*, Vol. 2, No. 3, pp. 408–429, 2001.

[52] Orna Kupferman and Moshe Y. Vardi. Safraless decision procedures. In *Proc. 46th Annual IEEE Symposium on Foundations of Computer Science*, pp. 531–542. IEEE Computer Society, 2005.

[53] Jan Křetínský and Javier Esparza. Deterministic automata for the (F, G)-fragment of LTL. In *Proc. 24th International Conference on Computer Aided Verification*, Vol. 7358 of *LNCS*, pp. 7–22. Springer, 2012.

[54] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM: Probabilistic model checking for performance and reliability analysis. *SIGMETRICS Perform. Eval. Rev.*, Vol. 36, No. 4, pp. 40–45, 2009.

[55] François Laroussinie and Nicolas Markey. Temporal logic with forgettable past. In *Proc. 17th Annual IEEE Symposium on Logic in Computer Science*, pp. 383–392. IEEE Computer Society, 2002.

[56] Timo Latvala, Armin Biere, Keijo Heljanko, and Tommi A. Junttila. Simple bounded LTL model checking. In *Proc. 5th International Conference on Formal Methods in Computer-Aided Design*, Vol. 3312 of *LNCS*, pp. 186–200. Springer, 2004.

[57] P. Madhusudan and P. S. Thiagarajan. Distributed controller synthesis for local specifications. In *Proc. 28th International Colloquium on Automata, Languages and Programming,*, Vol. 2076 of *LNCS*, pp. 396–407. Springer, 2001.

[58] Madjid Maidi. The common fragment of CTL and LTL. In *Proc. 41st Annual Symposium on Foundations of Computer Science*, pp. 643–652. IEEE Computer Society, 2000.

[59] Zohar Manna and Amir Pnueli. A hierarchy of temporal properties. In *Proc. 6th Annual ACM Symposium on Principles of Distributed Computing*, pp. 205–205. ACM, 1987.

[60] Rene Mazala. Infinite games. In *Automata, Logics, and Infinite Games*, Vol. 2500 of *LNCS*, pp. 23–38. Springer, 2001.

[61] Robin Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[62] Satoru Miyano and Takeshi Hayashi. Alternating finite automata on omega-words. *Theor. Comput. Sci.*, Vol. 32, pp. 321–330, 1984.

[63] 望月翔平, 島川昌也, 萩原茂樹, 米崎直樹. LTL 式から Büchi オートマトンへの高速な変換法. ソフトウェア工学の基礎 XIX, 日本ソフトウェア科学会 FOSE2012, 2012.

[64] Ryosei Mori and Naoki Yonezaki. Several realizability concepts in reactive objects. In *Proc. Information Modeling and Knowledge Bases IV: Concepts, Methods and Systems*, pp. 407–424. IOS Press, 1993.

[65] Ryosei Mori and Naoki Yonezaki. Derivation of the input conditional formula from a reactive system specification in temporal logic. In *Proc. Formal Techniques in Real-Time and Fault-Tolerant Systems*, Vol. 863 of *LNCS*, pp. 567–582. Springer, 1994.

[66] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proc. 38th Annual Design Automation Conference*, pp. 530–535. ACM, 2001.

[67] Benjamin C. Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. In *Proc. 8th Annual IEEE Symposium on Logic in Computer Science*, pp. 376–385. IEEE Computer Society, 1993.

[68] Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *Proc. 21st Annual IEEE Symposium on Logic in Computer Science*, pp. 255–264. IEEE Computer Society, 2006.

[69] Nir Piterman and Amir Pnueli. Synthesis of reactive(1) designs. In *Proc. Verification, Model Checking, and Abstract Interpretation*, Vol. 3855 of *LNCS*, pp. 364–380. Springer, 2006.

[70] Amir Pnueli. The temporal logic of programs. In *Proc. 18th Annual Symposium on Foundations of Computer Science*, pp. 46–57. IEEE Computer Society, 1977.

[71] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 179–190. ACM, 1989.

[72] Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. In *Proc. 16th International Colloquium on Automata, Languages and Programming*, Vol. 372 of *LNCS*, pp. 652–671. Springer, 1989.

[73] Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *Proc. 31st Annual Symposium on Foundations of Computer Science*, pp. 746–757 vol.2. IEEE Computer Society, 1990.

[74] Roni Rosner. *Modular Synthesis of Reactive Systmes.* PhD thesis, Weizmann Institute of Science, 1992.

[75] Shmuel Safra. On the complexity of omega-automata. In *Proc. 29th Annual Symposium on Foundations of Computer Science*, pp. 319–327. IEEE Computer Society, 1988.

[76] Sven Schewe and Bernd Finkbeiner. Bounded synthesis. In *Proc. 5th International Symposium on Automated Technology for Verification and Analysis*, Vol. 4762 of *LNCS*, pp. 474–488. Springer, 2007.

[77] Mary Sheeran, Satnam Singh, and Gunnar Stålmarck. Checking safety properties using induction and a SAT-solver. In *Proc. Third International Conference on Formal Methods in Computer-Aided Design*, Vol. 1954 of *LNCS*, pp. 108–125. Springer, 2000.

[78] Daniel Sheridan. Bounded model checking with SNF, alternating automata, and Büchi automata. *Electron. Notes Theor. Comput. Sci.*, Vol. 119, No. 2, pp. 83–101, 2005.

[79] 島川昌也, 萩原茂樹, 米崎直樹. 仕様の自動検証に適した LTL フラグメント–実現集合を表す決定性オートマトン構成の立場から–. 日本ソフトウェア科学会第 23 回大会講演論文集, 2006.

[80] 島川昌也, 萩原茂樹, 米崎直樹. リアクティブシステム仕様の強充足可能性判定問題の計算量について. 第 4 回システム検証の科学技術シンポジウム予稿集, pp. 21–29, 2007.

[81] 島川昌也, 萩原茂樹, 米崎直樹. SAT solver を用いたリアクティブシステム仕様の有界強充足可能性判定. 日本ソフトウェア科学会第 25 回大会講演論文集, 2008.

[82] 島川昌也, 萩原茂樹, 米崎直樹. 実現可能性判定コスト削減のための LTL 構文の制限とそれによる仕様の判定法. 日本ソフトウェア科学会第 26 回大会講演論文集, 2009.

[83] Masaya Shimakawa, Shigeki Hagihara, and Naoki Yonezaki. Complexity of checking strong satisfiability of reactive system specifications. In *Proc. International Conference on Advances in Information Technology and Communication*, pp. 42–51, 2012.

[84] Masaya Shimakawa, Shigeki Hagihara, and Naoki Yonezaki. Complexity of strong satisfiability problems for reactive system specifications. *IEICE Trans. Inf. & Syst.*, Vol. E96-D, No. 10, pp. 2187–2193, 2013.

[85] Masaya Shimakawa, Shigeki Hagihara, and Naoki Yonezaki. SAT-based bounded strong satisfiability checking of reactive system specifications. In *Proc. International Conference on Information and Communication Technology (ICT-EurAsia2013)*, Vol. 7804 of *LNCS*, pp. 60–70. Springer, 2013.

[86] Masaya Shimakawa, Shigeki Hagihara, and Naoki Yonezaki. Bounded strong satisfiability checking of reactive system specifications. *IEICE Trans. Inf. & Syst.*, Vol. E97-D, No. 7, pp. 1746–1755, 2014.

[87] João P. Marques Silva and Karem A. Sakallah. Grasp – a new search algorithm for satisfiability. In *Proc. 1996 IEEE/ACM International Conference on Computer-aided Design*, pp. 220–227. IEEE Computer Society, 1996.

[88] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, Vol. 32, No. 3, pp. 733–749, 1985.

[89] A. Prasad Sistla, Moshe Y. Vardi, and Pierre Wolper. The complementation problem for Büchi automata with appplications to temporal logic. *Theor. Comput. Sci.*, Vol. 49, No. 2–3, pp. 217–237, 1987.

[90] Heikki Tauriainen. On translating linear temporal logic into alternating and nondeterministic automata. Research Report A83, Helsinki University of Technology, Laboratory for Theoretical Computer Science, 2003.

[91] Wolfgang Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B)*, pp. 133–192. Elsevier and MIT Press, 1990.

[92] 冨田尭. 定量的性質を記述可能な拡張時間論理とそれを用いた仕様の検証手法に関する研究. 博士論文, 東京工業大学, 2013.

[93] 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 回大会講演論文集, 2013.

[94] Vaithinathan Vanitha, Kenji Yamashita, Kimiyuki Fukuzawa, and Naoki Yonezaki. A method for structuralisation of evolutional specifications of reactive systems. In *Proc. 3rd International Workshop on Intelligent Software Engineering (WISE3)*, pp. 30–38, 2000.

[95] Moshe Y. Vardi. An automata-theoretic approach to fair realizability and synthesis. In *Proc. 7th International Conference on Computer Aided Verification*, Vol. 939 of *LNCS*, pp. 267–278. Springer, 1995.

[96] Martin De Wulf, Laurent Doyen, Nicolas Maquet, and Jean-François Raskin. Antichains: Alternative algorithms for LTL satisfiability and model-checking. In *Proc. 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Vol. 4963 of *LNCS*, pp. 63–77. Springer, 2008.

[97] Noriaki Yoshiura. Decision procedures for several properties of reactive system specifications. In *Proc. International Symposium on*

Software Security - Theories and Systems (ISSS2003), Vol. 3233 of *LNCS*, pp. 154–173. Springer, 2004.

[98] Lintao Zhang, Conor F. Madigan, Matthew H. Moskewicz, and Sharad Malik. Efficient conflict driven learning in a boolean satisfiability solver. In *Proc. 2001 IEEE/ACM International Conference on Computer-aided Design*, pp. 279–285. IEEE Computer Society, 2001.

付録

ここでは，[7, 8] による簡易 n 階建てエレベータの仕様 Ele_n，及びその仕様に環境に関する仮定を付加した Ele_n^a を示す．

Ele_n は，以下に示す式集合 Φ のすべての要素の連言である．ここで，X は要求イベントの集合であり，Y は応答イベントの集合である．演算子 $\mathbf{U}_w$， $\mathbf{R}_s$ は，以下のような略記である．

$$f_1\ \mathbf{U}_w\ f_2 \equiv f_2\ \mathbf{R}(f_2 \vee f_1) \quad \text{［弱い until］}$$
$$f_1\ \mathbf{R}_s\ f_2 \equiv f_2\ \mathbf{U}(f_2 \wedge f1) \quad \text{［強い release］}$$

Ele_n^a は，Ele_n に「各階の呼び出しボタンは，無限に押し続けられることはない」という仮定を付加した仕様であり，

$$\mathbf{GF}\neg ReqBtn[1] \wedge \ldots \wedge \mathbf{GF}\neg ReqBtn[n] \rightarrow Ele_n$$

である．

$X = \{$
$ReqBtn[i](i = 1..n)$, //i 階の呼出しボタンが押された
$OpenBtn$, //「開く」ボタンが押された
$CloseBtn$ //「閉じる」ボタンが押された
$\}$

$Y = \{$
$Loc[i](i = 1..n)$, //リフトが i 階にある
$ReqL[i](i = 1..n)$, //リフトが i 階に行く要求がある
$Open$, //ドアが開いている
$Move$, //リフトが可動状態である
$OpenTimedOut$, //「開く」ボタンの時間切れ
$OpenReq$ //ドアを開く要求がある
$\}$

$\Phi = \{$

//リフトはどこかの階にある
$\Box(\bigvee_{1\leq i\leq n} Loc[i]),$

//リフトがある階にある場合，他の階にはない
$\Box(\bigwedge_{1\leq i\leq n} (Loc[i] \rightarrow \bigwedge_{j=1..n, i\neq j} \neg Loc[j])),$

//呼び出しボタンが押されればその階にいつか行く
//かつその要求が満たされるまで要求し続ける
$\Box(\bigwedge_{1\leq i\leq n} (ReqBtn[i] \rightarrow \mathbf{F}Loc[i] \wedge ReqL[i]\ \mathbf{U}_w(Loc[i] \wedge ReqL[i]))),$

//リフトのある階に要求があればドアを開く
//ドアが開いている間，リフトは停止
$\Box(\bigwedge_{1\leq i\leq n} (Loc[i] \wedge ReqL[i] \rightarrow Open \wedge Loc[i]\ \mathbf{U}_w Move)),$

//呼び出しボタンが押されるまで要求は出さない
$\Box(\bigwedge_{1\leq i\leq n} (Loc[i] \wedge Move \rightarrow \neg ReqL[i]\ \mathbf{U}_w ReqBtn[i]),$

//その階に要求がなければ，ドアは開かない
$\Box(\bigwedge_{1\leq i\leq n} (Loc[i] \wedge \neg ReqL[i] \rightarrow \neg Open)),$

//途中階を通る
($n \geq 3$ の場合)
$\Box(\bigwedge_{\substack{1\leq i\leq n-2\\ 3\leq j\leq n\\ i\leq j-2}} (Loc[i] \wedge ReqL[j] \rightarrow \bigwedge_{i+2\leq k\leq j} Loc[k-1]\mathbf{R}_s\neg Loc[k])),$

($n \geq 3$ の場合)
$\Box(\bigwedge_{\substack{1\leq i\leq n-2\\ 3\leq j\leq n\\ i\leq j-2}} (Loc[j] \wedge ReqL[i] \rightarrow \bigwedge_{i+2\leq k\leq j} Loc[k+1]\mathbf{R}_s\neg Loc[k])),$

//ドアの開閉とリフトの可動状態の関係
$\mathbf{G}(Open \rightarrow \neg Move\ \mathbf{U}_w \neg Open),$
$\mathbf{G}(\neg Open \rightarrow Move\ \mathbf{U}_w Open),$

//ドアの開放状態には時間制限がある
$\mathbf{G}(Open \rightarrow \mathbf{F}\,OpenTimedOut)$,

//時間制限内で “開く” ボタンを押せば
//ドアの開放を要求
$\mathbf{G}(OpenBtn \wedge \neg OpenTimedOut \rightarrow OpenReq)$,

//ドア開放時間制限が過ぎたらドアを閉める
$\mathbf{G}(OpenTimedOut \rightarrow \neg Open)$,

//開放要求がなくかつ “閉じる” ボタンが押されれば
//ドアを閉じる
$\mathbf{G}(CloseBtn \wedge \neg OpenReq \rightarrow \neg Open)$,

//ドア開放要求が真，可動状態が偽ならドアを開く
$\mathbf{G}(OpenReq \wedge \neg Move \rightarrow Open)$
}