博士学位論文

連携部分に着目した分割統治法および安全性プロパティ導出ルールを用いた複雑な組込みシステムに対するモデル検査適用手法

加藤　淳
Atsushi KATOH

慶應義塾大学大学院
システムデザイン・マネジメント研究科
後期博士課程

A doctoral course student of
Keio University,
Graduate School of System Design and Management

博士学位論文要旨

連携部分に着目した分割統治法および安全性プロパティ導出ルールを用いた複雑な組込みシステムに対するモデル検査適用手法

慶應義塾大学大学院
システムデザイン・マネジメント研究科
後期博士課程

加藤　淳

特定の機能を実現するために機械や機器に組み込まれるコンピュータシステム（組み込みシステム）において，その信頼性を向上させることは，安心・安全な社会を実現する上で重要な課題である．これは，近年，組込みシステムは，社会的なインフラストラクチャを始め，日常生活に深く浸透しており，その不具合が社会に大きな影響を与えるためである．一方，求められる機能が高度になることで，組込みシステムは複雑化している．その結果，放射線照射装置の事故，アリアン 5 の爆発事故など，社会に大きな影響を与える不具合が発生している．

複雑な組込みシステム（以降，システムという）の信頼性を向上させる技術として，形式手法の 1 つであるモデル検査が注目されている．モデル検査とは，仕様書などから抽出した検証対象の状態遷移が，検証対象として満たすべき性質（以降，プロパティという）を満足するか否かを，網羅的に検証する技術である．モデル検査を用いることにより，システム開発の早期の段階において，複雑な条件に起因する検証対象の不具合を検出することができる．このような複雑な条件に起因する不具合は，人手による検出が困難であるため，高い信頼性が求められるシステム開発において有用な技術である．

しかし，複雑なシステムに対するモデル検査の適用には，つぎに示す 2 つの課題がある．1 つ目の課題は，モデル検査が現実的な時間で終了しない状態爆発と

いう現象がしばしば発生することである．複雑なシステムは，それを構成する複数の要素で構築される．システムの構成要素およびその連携部分を考慮してモデル検査を実施する場合，システムの構成要素のすべてを対象に，一括してモデル検査を実施する方法が考えられる．その際，モデル検査では，各構成要素におけるすべての変数の組み合わせを網羅的に検証するため，状態の組み合わせが膨大になる．

2 つ目の課題は，具体的な安全性プロパティの導出において，導出すべき安全性プロパティに見逃しが生じる可能性や，具体化が不十分になる可能性が生じることである．モデル検査を実施する際には，例えば「列車がロードユーザ（自動車や歩行者）と衝突する」など，高次な複雑さを有するシステムの望ましくない事象から，モデル検査への適用が可能な具体的な安全性プロパティを導出する必要がある．複雑なシステムは，考慮すべき機能や条件が多く複雑であるため，具体的な安全性プロパティを系統立てて導出する必要がある．しかし，実際の導出は，モデル検査実施者の経験やスキルに基づくことが多い．

本研究では 1 つ目の課題を解決するために，システムの構成要素および構成要素の連携に関連する部分を，それぞれ個別にモデル検査する手法を提案する．連携に関連する仕様は，アーキテクチャ設計プロセスで作成するトレーサビリティマトリクスを用いて，構成要素の仕様書および構成要素間のインタフェースの仕様書から漏れなく抽出する．すべての構成要素に対して一括してモデル検査を実施するのではなく，構成要素および構成要素の連携部分に着目し，モデル検査を実施することで，状態爆発を回避する．本手法の評価実験において，状態爆発を発生させることなく，構成要素を組み合わせなければ発見できない不具合を検出することができた．

また 2 つ目の課題を解決するために，「Must Work Function（MWF）非起動」および「Must Not Work（MNWF）起動」の観点を用いて，モデル検査に適用可能な具体的な安全性プロパティを導出する手法を提案する．導出するプロパティの網羅性向上に寄与すると考えられる「MWF 非起動」および「MNWF 起動」の 2 つの観点を用いて，システムの望ましくない事象を具体化する．これらの観点に基づき，望ましくない事象の具体化を繰り返すことで，より具体的な事象を導出する．望ましくない事象の具体化の後，具体化した事象の否定（¬）を取ることで，具体的な安全性プロパティを導出する．本手法の評価実験において，導出する安全性プロパティの網羅性および具体性が向上することを確認

した.

本研究で取り上げた 2 つの課題解決を通して, 複雑な組込みシステムに対するモデル検査適用のハードルを下げることができた．それにより，モデル検査の研究領域の発展および産業界における組込みシステムの信頼性向上に貢献することができた.

Model Checking with Efficient Model Verification and Correct Safety Property Derivation for Complex Embedded Systems

Atsushi KATOH

A doctoral course student of
Keio University,
Graduate School of System Design and Management

Thesis Abstract

For embedded systems where computers are embedded into machines and equipment to realize specific functions, to improve their reliability is an important issue. Failures of embedded systems seriously influence society because nowadays embedded systems are widely propagated into our daily life. Moreover, while higher functions are desired for embedded systems, the systems are getting more complex. Some accidents such as irradiation equipment's malfunction and Ariane 5 rocket self-destruct happened because of their system's complexity. The failures influence society widely.

Model checking, one of formal methods, receives attention as a technology which improves reliability of complex embedded systems. Model checking is a technology which comprehensively verifies whether the target's state transitions extracted from its specifications satisfy the properties as a target should satisfy. On an early phase of the system development, model checking enables us to find failures of a target which are due to its complex conditions. Model checking is a very effective method in the system development which requires high reliability, because such failures due to complex conditions are difficult to find by hand.

However, there are two issues to apply model checking to complex systems. First, "state explosion" frequently happens and model checking does not finish in a realistic time. A complex system consists of several components. When we apply model checking to the system with considering the components and their coordination, we

target all the components and conduct model checking concurrently. Here, the number of state combination gets significantly large because model checking verifies all combinations of all components' variables comprehensively.

Second, deriving specific safety properties has the following two problems: some safety properties may be neglected, and some safety properties may be insufficiently specified. It is necessary to derive specific safety properties which can be applied for model checking from complex undesirable events in systems in order to perform model checking. Because complex systems have many functions and/or conditions to consider, it is necessary to derive specific safety properties systematically. However, deriving specific safety properties depends on experiences and skills of model checking conductors, and specific safety properties are not sometimes derived properly.

To solve the first issue, we propose a method to conduct model checking to each component and coordination of components in a system individually. The specifications of component coordination are thoroughly extracted from specification documents of components and interface specification documents between components using traceability matrix developed on architectural design phase. We avoid the state explosion not by applying model checking to all components concurrently, but by focusing on each component and component coordination and conducting model checking to them. We could find failures without state explosion in the evaluation of this method, and those failures could be found only by combining the components.

To solve the second issue, we propose a method for deriving comprehensive and specific safety properties that are applicable to model checking with the concepts of “Must Work Function (MWF) Inactivation” and “Must Not Work Function (MNWF) Activation”. An undesirable event in a system is embodied by these ideas of “MWF Inactivation” and “MNWF Activation”, which are expected to improve the comprehensiveness of the derived safety properties. The process of embodiment is conducted repeatedly. As a result, more comprehensive and specific events are derived. After deriving the embodied events from an undesirable event, safety properties are developed by negating (¬) the embodied events. We could confirm that the derived safety properties offer a significantly improved degree of comprehensiveness and specificity in the evaluation of this method.

With solving two issues taken up in this study, the hurdle to applying model checking

to complex embedded systems could be lowered. Therefore, we could contribute the growth of research area for model checking and boost the improvement of reliability on embedded systems in industry.

目　次

図表目次

第1章　序論

1.1　背景

組込みシステムの信頼性を向上させることは，安心・安全な社会を実現する上で重要な課題である．組込みシステムとは，特定の機能を実現するために機械や機器に組み込まれたコンピュータシステムである．近年，組込みシステムは，社会的なインフラストラクチャを始め，日常生活に深く浸透している．図 1-1 に，組込みシステムの搭載例を示す．鉄道，産業用ロボット，航空機などは，代表的な組込みシステムの搭載例である．一方で，機能の高機能化，多機能化により，組込みシステムは複雑化している．図 1-2 に，複雑な組込みシステムの例を示す．自動車の場合，基本性能，乗員の安全性や快適性，環境性を向上させるために，数十個のコンピュータが車体に搭載され，ネットワークを介して相互に連携している．組込みシステムが複雑化していることで，検証の困難さによる不具合が多発している．放射線照射装置の事故[1]，アリアン 5 の爆発事故[2]など，その不具合が社会に大きな影響を与えている．

図 1-1　組込みシステムの搭載例

複雑な組込みシステム（以降，システムという）の信頼性を向上させる技術として，形式手法[3]の 1 つであるモデル検査[4]が注目されている．モデル検査とは，仕様書などから抽出した検証対象の状態遷移が，検証対象として満たすべき性質（以降，プロパティという）[5]を満たすか否かを，網羅的に検証する技術である．モデル検査は，計算機上で動作するモデル検査ツールにより実現される．システム開発の上流工程において，システムの仕様などに対して適用される．モデル検査を用いることにより，システム開発の早期の段階において，複雑な条件に起因する検証対象の不具合を検出することができる．複雑な条件に起因する不具合は，人手による検出が困難である．そのため，モデル検査は，高い信頼性が求められる組込みシステム開発において，有用な技術である．

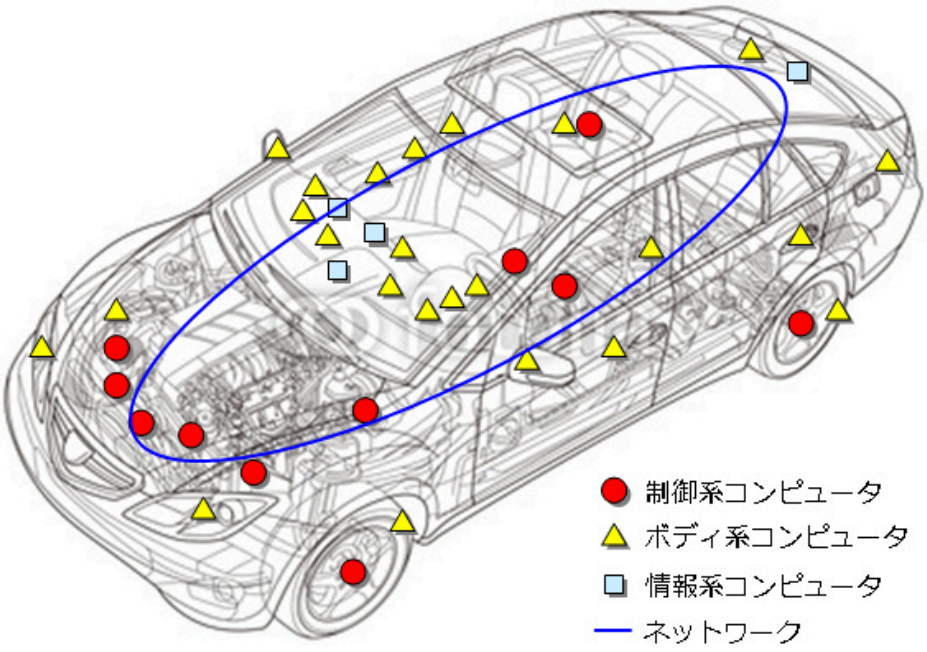


図 1-2　複雑な組込みシステムの例

1.2　課題

複雑なシステムに対してモデル検査を適用する場合，モデル検査適用者の視点から見て，つぎに示す 2 つの課題がある．

複雑なシステムに対するモデル検査適用時の 1 つ目の課題は，モデル検査ができない，もしくは，現実的な時間では終了しないことである．図 1-3 に，複雑なシステムに対するモデル検査適用時の 1 つ目の課題を示す．複雑なシステムは，それを構成する複数の要素で構築される．システムの構成要素は，それぞれが連携しながらシステムの機能を実現する．モデル検査では，各構成要素におけるすべての変数の組み合わせを網羅的に検証する．そのため，複雑なシステムを一括してモデル検査する場合，状態数が膨大になり，計算機のメモリ資源が枯渇する状態爆発という現象がしばしば発生する．

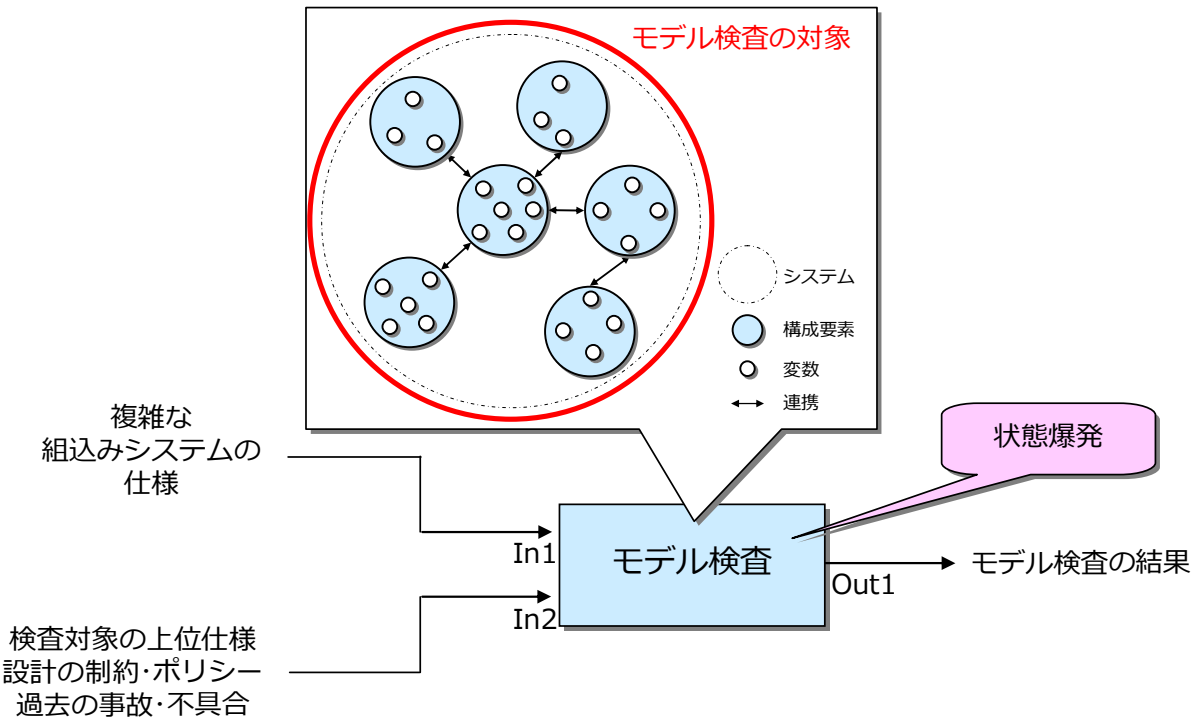


図 1-3　複雑なシステムへのモデル検査適用時の課題 1

複雑なシステムに対するモデル検査適用時の 2 つ目の課題は，安全性プロパティの導出が困難なことである．図 1-4 に，複雑なシステムに対するモデル検査適用時の 2 つ目の課題を示す．モデル検査を実施する際には，例えば，「列車がロードユーザ（自動車や歩行者）に衝突する」など，高次な複雑さを有するシステムの望ましくない事象から，モデル検査への適用が可能な具体的な安全性プロパティを導出する必要がある．複雑なシステムでは，具体的な安全性プロパティの導出時において，考慮すべき機能や条件が複雑である．そのため，導出すべき安全性プロパティに見逃しが生じる可能性や，具体化が不十分になる可能性がある．

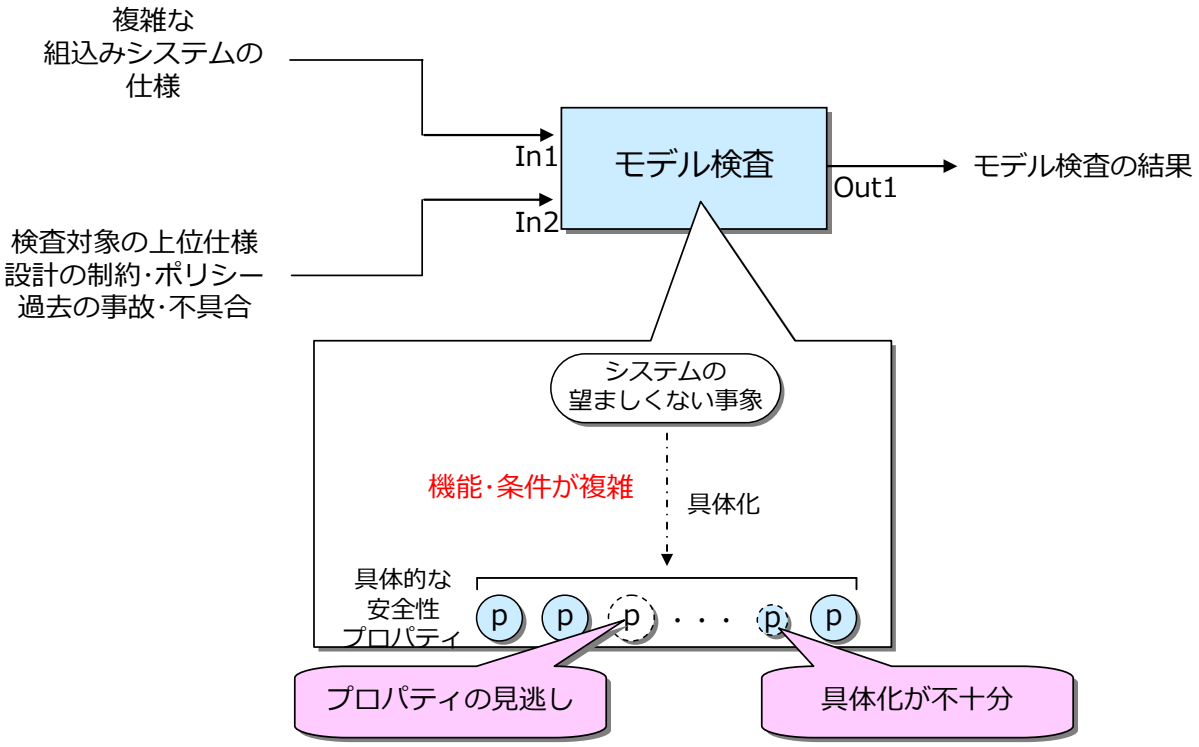


図 1-4　複雑なシステムへのモデル検査適用時の課題 2

1.3　研究の目的

本研究の目的は，1.2 で述べた課題 1 および課題 2 を解決することで，複雑なシステムに対してモデル検査の適用を可能にする手法を確立することである．

図 1-5 に，課題 1 を解決する手法を示す．課題 1 を解決するために，システムの構成要素および構成要素の連携に関連する部分を，それぞれ個別にモデル検査する．構成要素の連携部分をモデル検査する際に必要となる，連携部分の仕様は，システム開発のアーキテクチャ設計プロセス[6][7][8][9]で作成するトレーサビリティマトリクスを用いて，構成要素の仕様書および構成要素間のインタフェースの仕様書から漏れなく抽出する．システムを構成するすべての要素を対象に，一括してモデル検査を実施するのではなく，構成要素および構成要素の連携部分に着目して，個別にモデル検査を実施する．この分割統治法（Divide and Conquer）に基づくモデル検査により，状態爆発を回避しながら，システム全体を検証する．

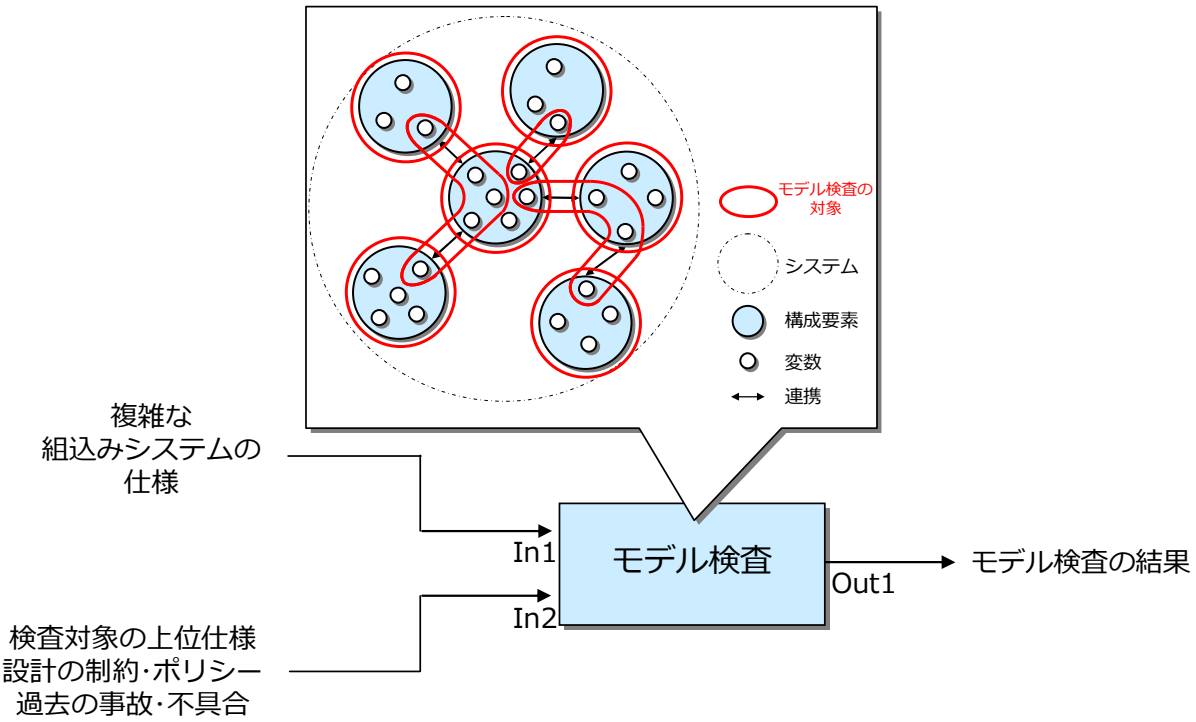


図 1-5　課題 1 に対する提案

図 1-6 に, 課題 2 を解決する手法を示す. 課題 2 を解決するために,「Must Work Function（以降，MWF という）非起動」および「Must Not Work（以降，MNWF という）起動」の観点を用いて，モデル検査に適用可能な具体的な安全性プロパティを導出する．MWF とは，その意図しない非起動が，システムの望ましくない事象を引き起こす可能性がある機能である[10]．MNWF とは，その意図しない起動が，システムの望ましくない事象を引き起こす可能性がある機能である[10]．導出するプロパティの網羅性向上に寄与すると考えられる「MWF 非起動」および「MNWF 起動」の 2 つの観点を用いて，システムの望ましくない事象を具体化する．また，これらの観点に基づき，望ましくない事象の具体化を繰り返すことで，より具体的な事象を導出する．望ましくない事象の具体化の後，具体化した事象の否定（¬）を取ることで，具体的な安全性プロパティを導出する．この安全性プロパティ導出ルールにより，モデル検査における安全性プロパティの抜けや具体化が不十分になることを防ぐ．

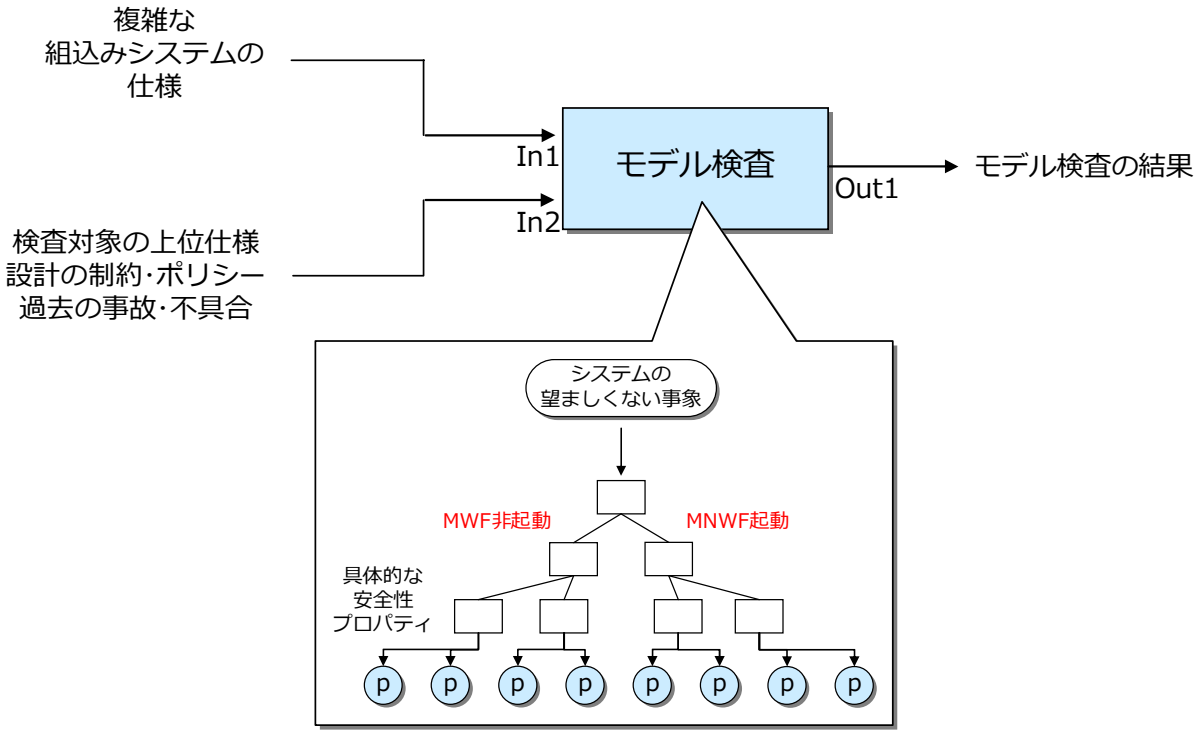


図 1-6　課題 2 に対する提案

図 1-7 に, 本論文で提案する, 複雑なシステムに対するモデル検査手法を示す. 図 1-7 におけるモデル検査の下部が，課題 1 に対する提案手法である．図 1-7 におけるモデル検査の上部が，課題 2 に対する提案手法である．モデル検査の検証対象であるシステムの望ましくない事象から，具体的な安全性プロパティを導出する．導出した具体的な安全性プロパティを基に，構成要素および構成要素間の連携部分をそれぞれ個別にモデル検査する．それにより，複雑なシステムへの効果的なモデル検査を実現し，複雑なシステムの信頼性を向上させる．

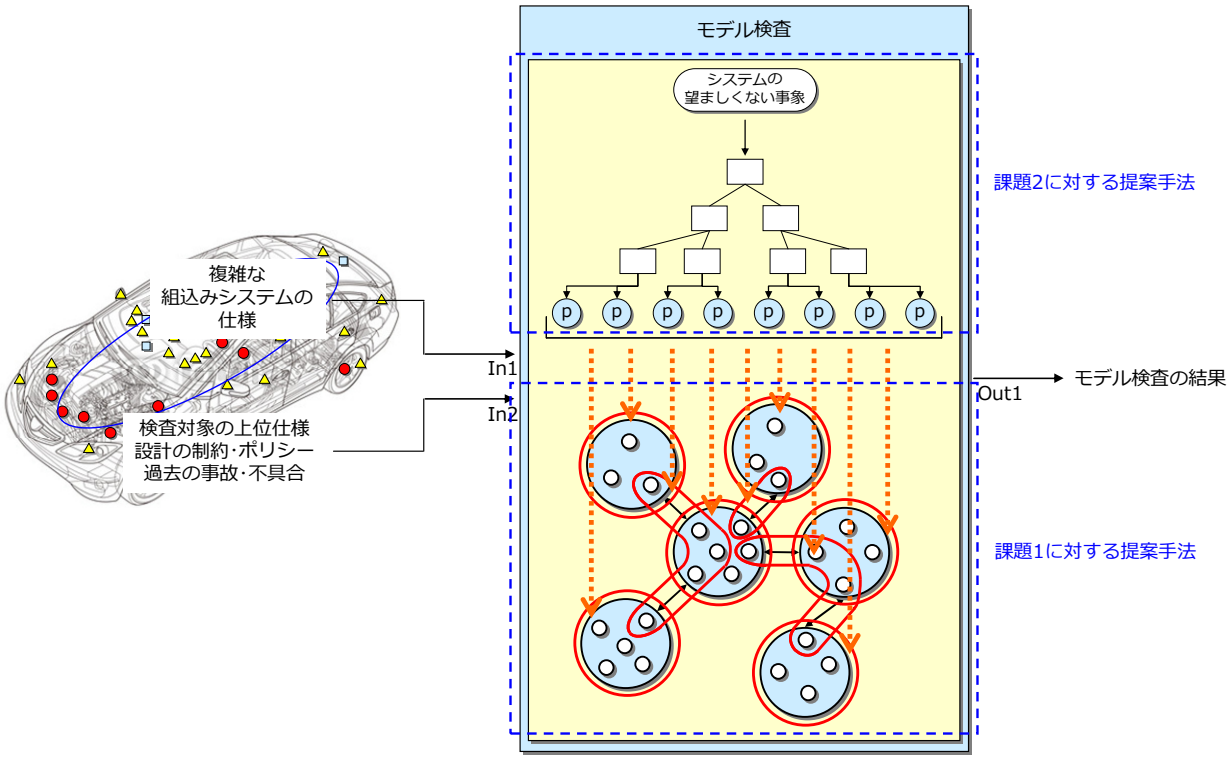
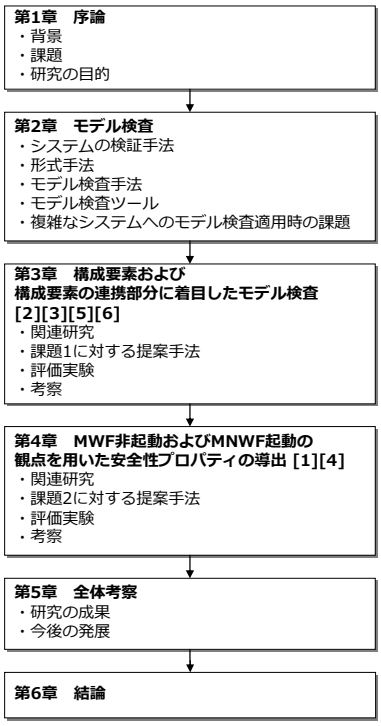


図 1-7　複雑なシステムに対するモデル検査手法

1.4　論文の構成

図 1-8 に，本論文の構成を示す．第 2 章において，モデル検査技術を解説する．第 3 章において，課題 1 を解決する，構成要素および構成要素の連携部分に着目したモデル検査手法を述べる．第 4 章において，課題 2 を解決する，MWF 非起動および MNWF 起動の観点を用いた，モデル検査における安全性プロパティの導出手法を述べる．第 5 章において，本論文で提案する，複雑なシステムに対するモデル検査手法の成果および今後の発展を考察する．第 6 章において，本論文をまとめる．



[1] Atsushi Katoh, Shinichiro Haruyama, Naohiko Kohtake and Yoshiaki Ohkami:
Model Checking of Safety Properties for Complex Systems Using MWF Inactivation and MNWF Activation,
International Journal of Computer Science and Mobile Computing, Vol.2, No.12, pp.26-39 (2013)

[2] 加藤淳，浦郷正隆，狼嘉彰：
構成要素の連携動作に着目した産業用ロボットの開発：
IEEE 1220におけるアーキテクチャ設計プロセスおよびUPPAALによるモデル検査を融合した開発手法の提案，
計測自動制御学会産業論文集，Vol.10，No.5，pp.37-49（2011年）

[3] 加藤淳，浦郷正隆，狼嘉彰：
複雑システムの信頼性を向上させる開発手法：アーキテクチャ設計手法とモデル検査の融合，
Synthesiology，Vol.3，No.3，pp.197-212（2010年）

[4] 加藤淳，松本充広，春山真一郎：
MWF/MNWFに基づくFTAとSpec Patternsによるモデル検査式の導出，
情報処理学会論文誌プログラミング（PRO），Vol.5，No.4，pp.40（2012年）

[5] 加藤淳，神武直彦，春山真一郎，狼嘉彰：
モデル検査を用いて組込みシステムにおける
ソフトウェアとハードウェアの協調動作に関する要求仕様の不整合を検出する手法，
組込みシステムシンポジウム2009，IPSJ Symposium Series Vol. 2009，pp.65-70（2009年）

[6] 加藤淳，狼嘉彰：
FPGAとソフトウェアにおける協調動作の整合性に関する評価手法の提案，
情報処理学会 第163回ソフトウェア工学研究会研究報告，Vol.2009，No.31，pp.105-112（2009年）

図 1-8　論文の構成

第2章　モデル検査

本章では，システムの検証手法および形式手法を解説し，モデル検査の技術的な位置付けを明確にする．そして，システムへの適用の観点からモデル検査を解説する．また，複雑なシステムに対してモデル検査を適用する際の，モデル検査適用者の視点から見た課題を述べる．

2.1　システムの検証手法

図 2-1 に，システムの検証手法とその適用工程を示す．システムの検証手法には大きく，テスティング[11]，レビュー[12]，シミュレーション[13]，形式手法[3]がある．表 2-1 に，それぞれの検証手法の特徴を示す．

テスティングは，特定の入力を与えて実システムを動作させ，実システムの出力を確認することにより，不具合を検出する方法である．システム開発の下流工程である構成要素試験，システム試験，受入試験で適用される代表的な検証手法である．テスティングにより，実システムの実際の動作を確認しながら，不具合の有無を確認することができる．一方で，実システムへの入力データによっては，不具合を検出することができない．また，その適用が下流工程であるため，検出した不具合の改修時には，開発の手戻りが発生する．

レビューは，開発工程で作成される，システムの成果物へのドキュメントレビューにより，不具合を検出する方法である．システム開発の上流工程であるシステム要求，システム設計，構成要素設計，構成要素実装で適用される代表的な検証手法である．レビューにより，システム開発の早期において，検証対象の不具合の有無を確認することができる．一方で，人手による検証につき，複雑な条件に起因する検証対象の不具合を検出することは困難である．

シミュレーションは, 検証対象であるシステムおよびシステムの周辺環境を計算機上で模擬する．そして，特定の入力に対するシステムの出力を計算機上で確認することによって，不具合を検出する方法である．システム開発の上流工程であるシステム要求，システム設計，構成要素設計，構成要素実装で適用される．シミュレーションにより，周辺環境が存在しないシステム開発の早期において，実システムの動作に近い振舞いを計算機上で確認しながら，検証対象の不具合の有無を確認することができる．一方で，テスティングと同様に，入力データによっては，不具合を検出することができない．また，検証対象であ

るシステムや，その周辺環境を計算機上で模擬するため，コストが掛かる．

形式手法は，開発工程で作成されるシステムの成果物を論理学，集合論，代数学などを用いて形式的に記述し，数学的な手段により不具合を検出する方法である．システム開発の上流工程であるシステム要求，システム設計，構成要素設計，構成要素実装で適用される．形式手法により，システム開発の早期において，検証対象の不具合（成果物の曖昧さ，不整合など）の有無を確認することができる．また，設定した観点において，検証対象を自動的かつ網羅的に検証することができる．一方で，成果物を形式的に記述する必要があるため，コストが掛かる．

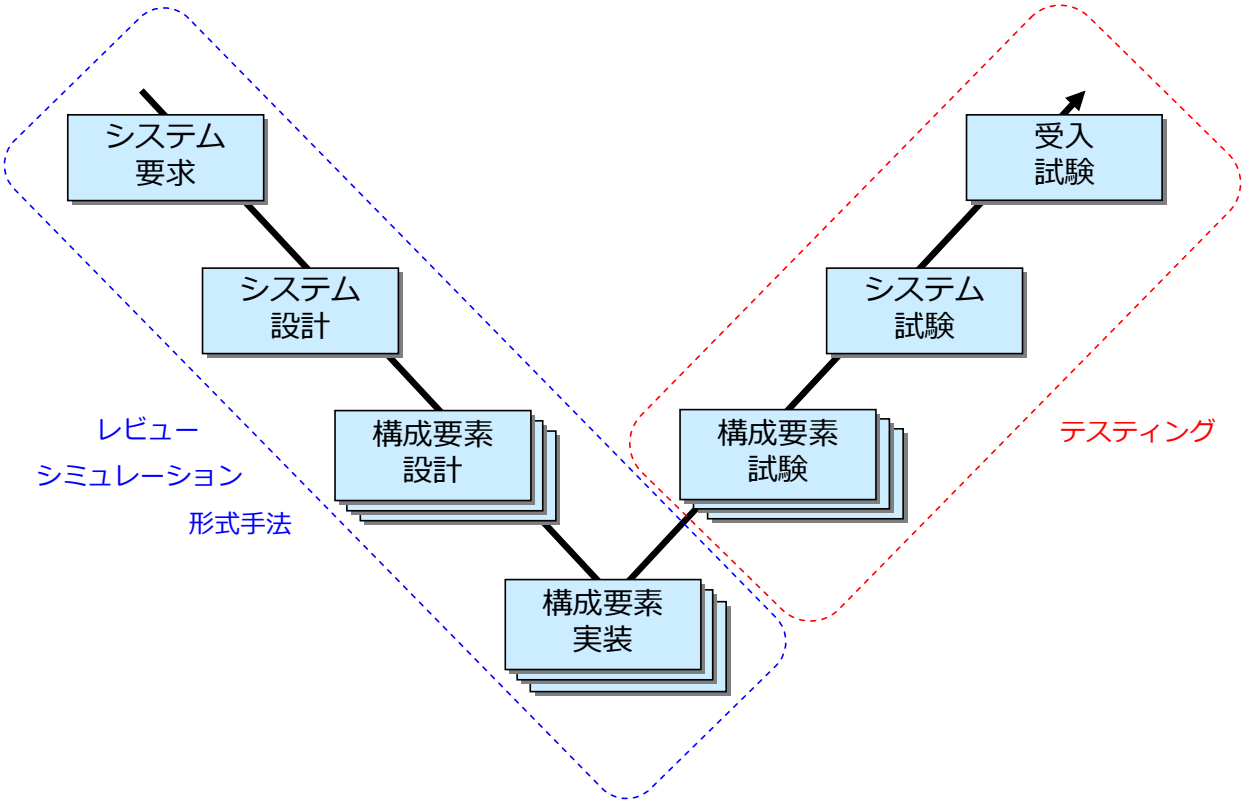


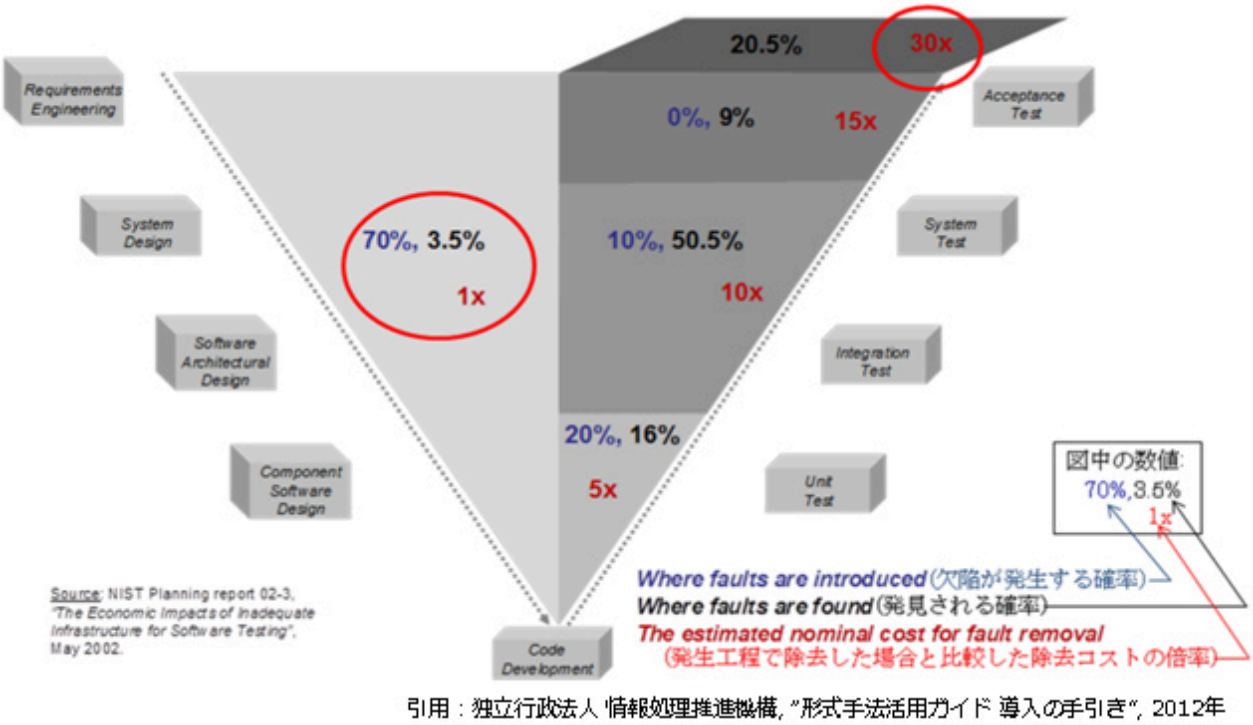
図 2-1　システムの検証手法と適用工程

表 2-1　システムの検証手法の特徴

検証技術	適用工程	概要	長所	短所
テスティング	・構成要素試験 ・システム試験 ・受入試験	特定の入力データに対して 実システムを動作させることで 不具合を検出	・実システムの動作を 確認しながら 不具合の検出が可能	・入力データによっては 不具合を検出することが できない ・不具合の改修時に 開発の手戻りが発生
レビュー	・システム要求 ・システム設計 ・構成要素設計 ・構成要素実装	開発工程の成果物に対する ドキュメントレビューにより 不具合を検出	・開発の上流工程で 不具合の検出が可能	・複雑な条件に起因する 不具合を検出することが 困難
シミュレーション		システム，その周辺環境を 計算機上に模擬し、 特定の入力データに対する システムの動作を計算機上で 確認することで不具合を検出	・開発の上流工程で 不具合の検出が可能 ・実システムの 動作に近い振舞いを 確認しながら 不具合の検出が可能	・入力データによっては 不具合を検出すること ができない ・システム、その周辺環境 を計算機上に模擬する ことにコストが掛かる
形式手法		開発工程の成果物を形式的に 記述し、数学的な手段で 不具合を検出	・開発の上流工程で 不具合の検出が可能 ・成果物の曖昧さ， 不整合などを検出可能 ・設定した観点において 自動的，網羅的な 検証が可能	・成果物を形式的に 記述することにコストが 掛かる

近年，システム開発における開発データの集計，分析により，システム開発の上流工程における検証の重要性が改めて認識されている[14]．図 2-2 に，システム開発の一つであるソフトウェア開発の，開発工程と不具合の発生・発見確率および不具合の改修コストの関係を示す[15]．ソフトウェア開発の各工程において，不具合が発生する確率（青字），不具合が発見される確率（黒字），上流工程の不具合を上流工程で改修した場合と比較した，改修コストの倍率（赤字）を示している．上流工程において不具合の 70%が発生しているものの，上流工程で発見される不具合は 3.5%である（図 2-2 における左の赤丸）．また，上流工程の不具合を保守段階で改修した場合，上流工程でそれらを改修した場合の 30 倍のコストを要する（図 2-2 における右の赤丸）．

図 2-2 から，システム開発の早期において，システムの品質を作り込むことの重要性が見て取れる．その方法として，形式手法が注目されている．



引用：独立行政法人 情報処理推進機構, "形式手法活用ガイド 導入の手引き", 2012年

図 2-2　ソフトウェア開発における不具合の発生・発見確率および改修コスト

2.2 形式手法

図 2-3 に，形式手法の種類を示す．形式手法には，形式仕様記述（Formal Specification）および形式検証（Formal Verification）がある[3]．モデル検査は，形式検証の一つである．表 2-2 に，それぞれの形式手法の特徴を示す．

形式仕様記述は, 形式的に定義されたシンタックスおよびセマンティクスを持つ言語を用いて，システムの機能的・時間的な振る舞い，性能，内部構造などを記述する手法である．システムの要件や設計の曖昧さを排除し，仕様を厳密に作成する．それにより，システム開発の上流工程における不具合の混入を防ぐ．形式仕様記述には，モデル規範と性質規範がある．モデル規範では，論理学や集合論を用いて仕様を記述する．モデル規範型の代表的な技術・ツールとして，Z 記法[16]，VDM[17]，B メソッド[18]や Event-B[19]といった B ファミリなどがある．性質規範では，代数学を用いて仕様を記述する．性質規範型の代表的な技術・ツールとして，OBJ[20]や CafeOBJ[21]といった OBJ ファミリ，LOTOS[22]，CSP[23]などがある．

形式検証は，形式的に記述されたシステムの仕様に誤りがないかどうかを，数学的な証明手段により保証する手法である．形式検証には，定理証明とモデル検査がある．定理証明では，検証対象が与えられた性質を満たすことを，推論規則を用いて検証者と対話的に証明する．述語論理などの形式的体系を用いて証明するため，本質的に無限の状態を含む検証対象を取り扱うことができる．しかし, 証明は検証者が行う必要があり，計算機による自動検証は困難である．定理証明の代表的な技術・ツールとして，coq[24]，HOL[25]，Isabelle[26]などがある．モデル検査は，計算機の支援を前提としている．モデル検査では，検証対象の状態遷移が，与えられたプロパティを満たすか否かを網羅的に自動検証する．モデル検査は，計算機能力の向上や検証アルゴリズムの革新により，組込みシステムなどの並行システムに対する実用的な自動検証技術になりつつある．また，与えられたプロパティを検証対象が満たさない場合，モデル検査ツールから反例が出力される．反例には，不具合箇所を特定する情報が含まれる．そのため，モデル検査を用いることで，不具合の効率的な検出が可能になる．モデル検査の代表的な技術・ツールとして，SPIN[27]，SMV[28]や NuSMV[29]といった SMV ファミリ，UPPAAL[30]などがある．

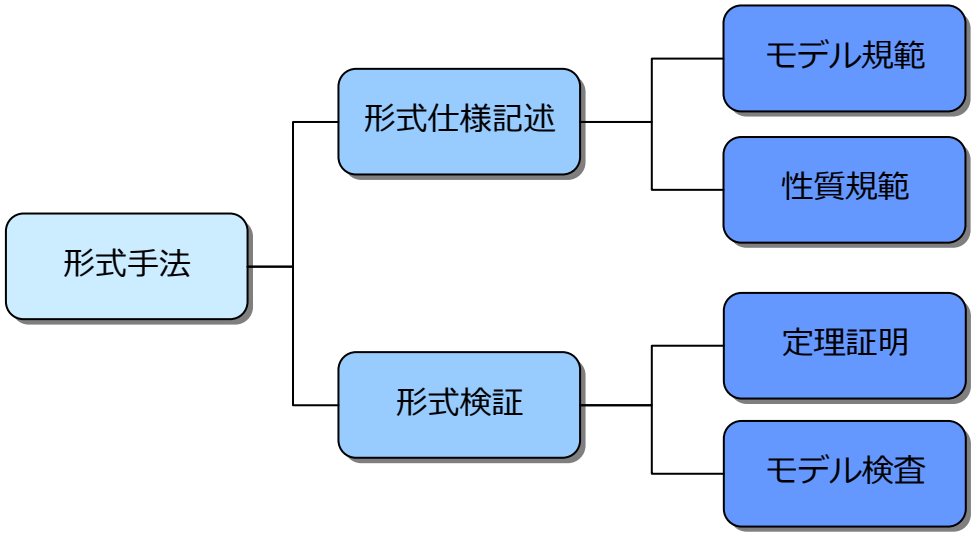


図 2-3　形式手法の種類

表 2-2　形式手法の特徴

形式手法		特徴		技術･ツール(一部)
大分類	小分類			
形式仕様記述	モデル規範	・形式的に定義されたシンタックスおよびセマンティクスを持つ言語を用いて，システムの機能的･時間的な振る舞い，性能，内部構造などを記述	・論理学、集合理論を用いて仕様を記述	Z記法 VDM Bファミリ
	性質規範		・代数学を用いて仕様を記述	OBJファミリ LOTOS CSP
形式検証	定理証明	・形式的に記述されたシステムの仕様に誤りがないかどうかを，数学的な証明手段により保証 ・計算機の支援を前提	・検証対象が与えられた性質を満たすことを推論規則を用いて検証者と対話的に証明	coq HOL Isabelle
	モデル検査		・検査対象の状態遷移が，与えられた性質を満たすか否かを網羅的に自動検証 ・反例による不具合の効率的な検出	SPIN SMVファミリ UPPAAL

図 2-4 に，形式仕様記述と形式検証の関係を示す．形式手法は，システム開発における段階的な詳細化という開発スタイルを前提に研究が進められてきた[31][32][33]．形式仕様記述を用いて，非形式的に表現されたシステムの仕様書などを基に，形式的な仕様を作成する．形式的な方法によって，形式的な仕様を段階的に詳細化・洗練化する．それらを繰り返すことで，システムの設計や実装を行う．形式検証は，詳細化・洗練化前後の各々の仕様記述の整合性確認，詳細化・洗練化の正しさを確認することに用いる．

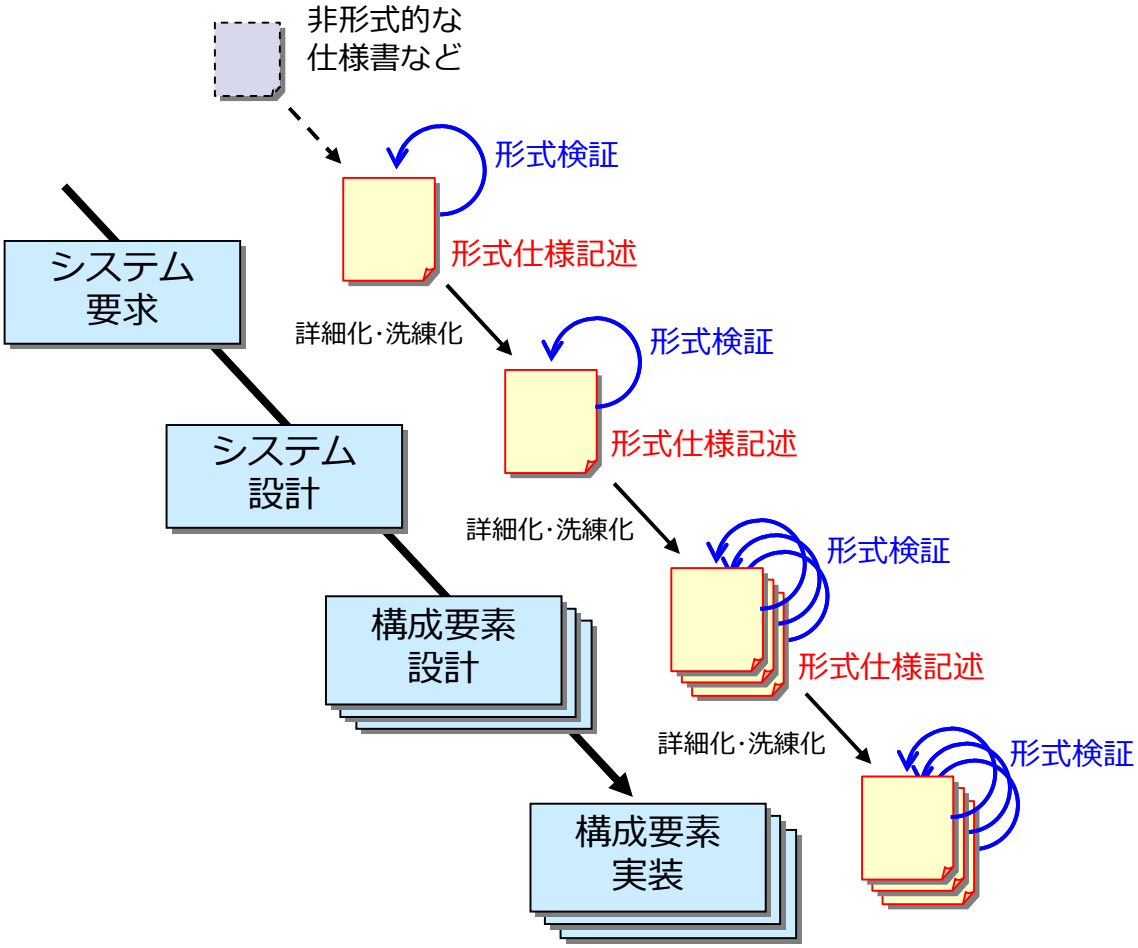


図 2-4　形式仕様記述と形式検証の関係

形式手法は，産業界から注目を集めている．その一因として，産業機器を対象とする IEC-61508[34]，自動車向けの ISO-26262[35]，システム・セキュリティに関する Common Criteria[36]などの機能安全に関する国際規格では，システム開発における形式手法，および，その支援ツールの利用が推奨されている．表 2-3 に，IEC-61508 における形式手法の推奨の一例を示す[37]．Safety Integrity Level（以降，SIL という）とは，システムの安全性能を表す尺度である．SIL1 から SIL4 までの 4 段階が定められている．SIL4 が最高の水準である．

表 2-3　IEC-61508 における形式手法の推奨の一例

Software safety requirements specification				
Technique/Measure	SIL1	SIL2	SIL3	SIL4
Computer-aided specification tools	R	R	HR	HR
Semi-formal methods	R	R	HR	HR
Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	–	R	R	HR
Software design and development: detailed design				
Technique/Measure	SIL1	SIL2	SIL3	SIL4
Semi-formal methods	R	HR	HR	HR
Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	–	R	R	HR
Computer-aided design tools	R	R	HR	HR

（R: Recommended，HR: Highly Recommended）

2.3 モデル検査手法

2.3.1 モデル検査のプロセス

図 2-5 に，モデル検査のプロセスを示す．モデル検査は，つぎに示す 4 つの工程で構成される．

工程 1. 状態遷移モデルの作成 : 検証対象の仕様から，検証対象の状態遷移を抽出する．適用するモデル検査ツールの表現形式に従い，抽出した検証対象の状態遷移をモデル化する．

工程 2. プロパティの導出 : 検証対象の上位仕様, 設計の制約・ポリシー，過去の事故・不具合などから，非形式的な形でプロパティを抽出する．Linear Temporal Logic（以降，LTL という）[38] や Computational Tree Logic（以降，CTL という）[38]といった時相論理[38]に従い，検査対象のプロパティを形式的に表現する．

工程 3. モデル検証 : 計算機上のモデル検査ツールに対して，状態遷移モデルおよびプロパティを入力し，モデル検証を実施する．モデル検証では, 状態遷移モデルがプロパティを満たすか否かを確認するために，状態遷移モデルの状態空間（状態遷移モデルが有する状態および状態遷移パス）が網羅的に探索される．モデル検証が終了した場合，モデルとプロパティの適合結果が出力される．適合結果が不適合の場合, プロパティが成り立たない状態に至るまでの状態遷移が反例として出力される．

工程 4. 検証結果の分析 : モデル検証の適合結果を分析する．適合結果が不適合の場合は，反例を分析する．反例の分析の結果，モデルおよびプロパティに誤りがない場合は, モデルの基になった検証対象の仕様に誤りがあることを意味する．

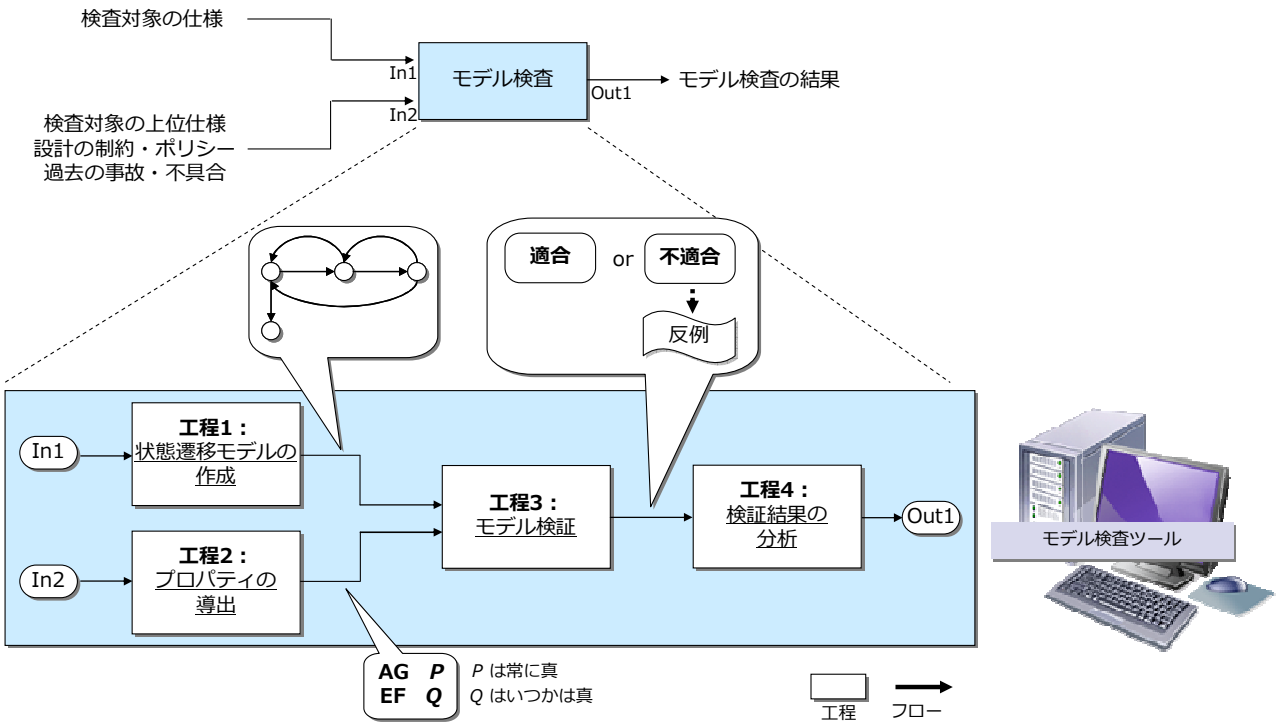


図 2-5　モデル検査のプロセス

モデル検査は，検証対象の正しさの証明というよりも，システム開発の上流工程における不具合の検出を目的として使用される場合が増えている．検証の対象を状態遷移に限定し，状態空間の網羅的探索に特化した自動解析を行う．このような，検証対象の限定や自動検証を前提とした形式手法の適用方法を，Light-weight Formal methods という[31].

モデル検査の適用では，不具合を検出することの他に，つぎに示す 2 つの効果が期待できる．1 つ目の効果は，反例を用いることで，不具合を検出する際の工数を削減できる可能性がある．テスティングやシミュレーションを適用し，確認内容が不適合となった場合，不適合に結びつく原因を複数仮定し，原因を分析する必要がある．その際，原因特定までに大きな労力を要することがある．モデル検査の場合，自動で出力される反例を用いて，不適合の発生過程を遡ることができる．それにより，不適合を引き起こす原因を効率よく特定することが可能である．2 つ目の効果は，モデル検査を実施する場合，モデル化という仕様の形式化を通して，検証対象である仕様の誤りを検出できる可能性がある．状態遷移モデルの作成においては，状態の遷移条件や遷移先をモデル中に明確に定義する必要がある．その際，検証者は，仕様書における遷移条件や遷移先の抜け，曖昧な定義に気付くことができる．

2.3.2 時相論理によるプロパティの表現

システム開発においてモデル検査を適用する目的は, 以下の性質を確認することに他ならない.

- システムにとって望ましい事象がいつかは起こる
- システムにとって望ましくない事象が決して起こらない

前者の性質を活性プロパティ，後者の性質を安全性プロパティという[5]．産業用ロボットの場合，システムにおける望ましい事象の例として,「ロボットシステムが運用者の負荷を低減する」が挙げられる．それに対応する活性プロパティは,「ロボットシステムが運用者の負荷を低減すること」である．また，システムにおける望ましくない事象の例として,「ロボットシステムが暴走する」が挙げられる．それに対応する安全性プロパティは,「ロボットシステムが暴走しないこと」である．

モデル検査のプロパティは，時相論理[38]などにより形式的に表現される．時相論理とは，時間の側面から問題を理解し表現するための規則と表記法の体系である．時相論理には，大きく LTL[38]および CTL[38]がある．

表 2-4 に，LTL を示す．LTL の時相演算子には，N : **n**ext（もしくは X : ne**x**t），G : **g**lobally，F : in the **f**uture，U : **u**ntil，R : **r**elease がある． ϕ および ψ を命題とした場合，「N ϕ」もしくは「X ϕ」は，「ϕ は次の状態で真である」を表す．「G ϕ」は，「ϕ はその後常に真である」を表す．「F ϕ」は，「ϕ はその後いずれかの時点で真となる」を表す．「ψ U ϕ」は，「ϕ は現在または将来のある時点で真であり，かつ，ψ はその時点まで真である（ただし，その時点以降 ψ は真になるとは限らない)」を表す．「ψ R ϕ」は，「現在または将来のある時点で ψ が真ならば，その直前まで ϕ は真である（ただし，その後 ϕ は真ではない)」を表す．「ψ R ϕ」は，「ϕ が真となることがない場合，ϕ は真のままである」を含む．

表 2-4　LTL

時相演算子	時相論理式	意味
N (**n**ext) or **X** (ne**x**t)	**N** φ or **X** φ	φ は次の状態で真である．
G (**g**lobally)	**G** φ	φ はその後常に真である．
F (in the **f**uture)	**F** φ	φ はその後いずれかの時点で真となる．
U (**u**ntil)	ψ **U** φ	φ は現在または将来のある時点で真であり， かつ，ψ はその時点まで真である． （その時点以降 ψ は真になるとは限らない）
R (**r**elease)	ψ **R** φ	現在または将来のある時点でψが真ならば， その直前までφは真である． （その後 φ は真ではない） ψ が真となることがない場合， φ は真のままである．

※ φ，ψは命題

図 2-6 に，LTL の概念を図示する．

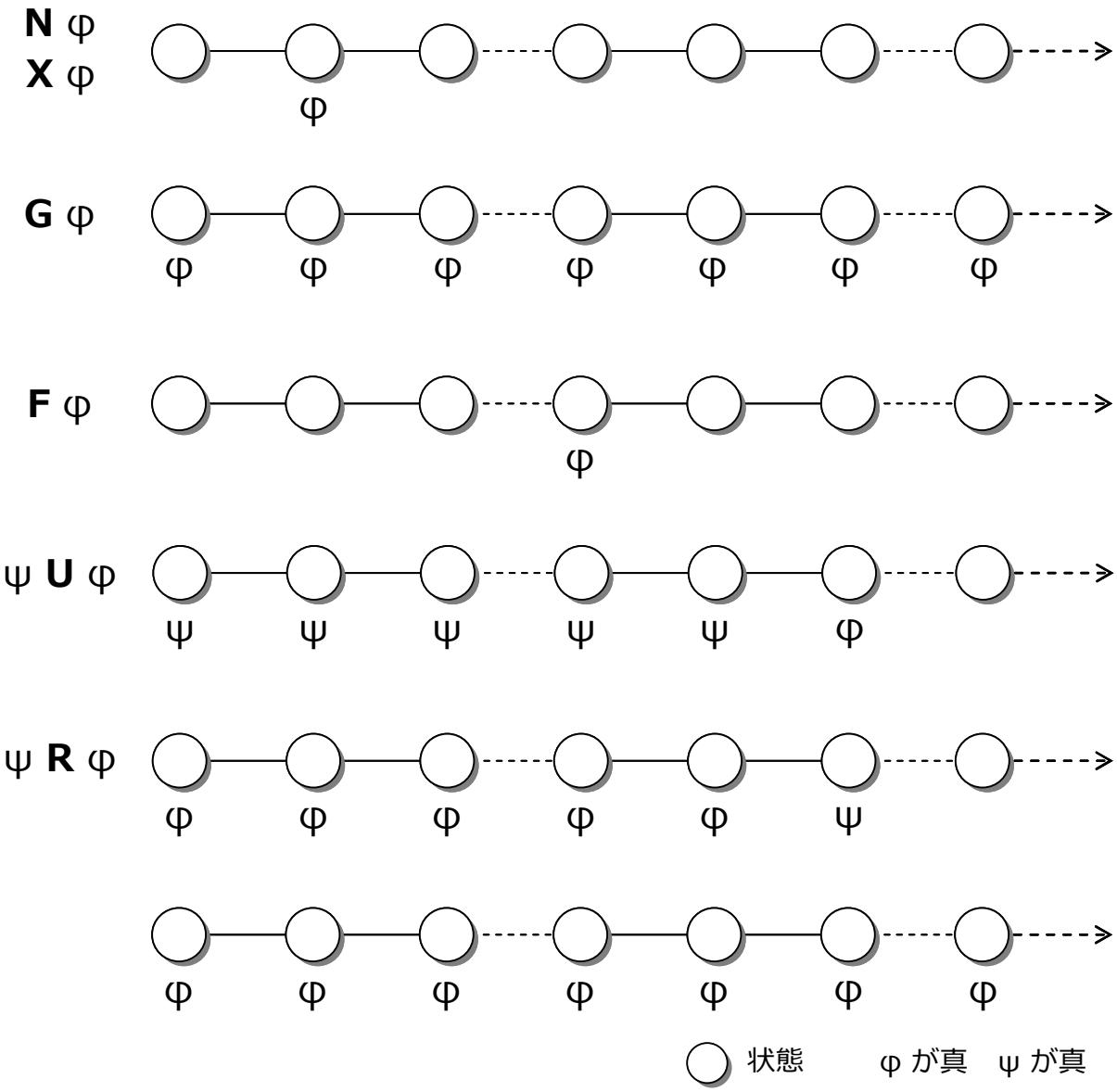


図 2-6　LTL の概念

表 2-5 に，CTL を示す．CTL の時相演算子には，経路演算子と状態演算子がある．経路演算子には，A : **a**ll，E : **e**xists がある．φおよびφを命題とした場合，「A φ」は，「現在の状態から分岐する全ての経路においてφが真である」を表す．「E φ」は，「現在の状態から分岐する経路のうち少なくとも 1 つの経路でφが真である」を表す. 状態演算子には, N : **n**ext（もしくは X : ne**x**t）, G : **g**lobally, F : in the **f**uture，U : **u**ntil，W : **w**eak until がある．「N φ」もしくは「X φ」は，「φは次の状態で真である」を表す．「G φ」は，「φはその後常に真である」を表す.「F φ」は,「φはその後いずれかの時点で真となる」を表す.「φ U φ」は,「φは現在または将来のある時点で真であり，かつ，φはその時点まで真である（ただし，その時点以降φは真になるとは限らない)」を表す．「φ W φ」は,「現在または将来のある時点でφが真ならば，その直前までφが真である」を表す．「φ W φ」は,「φが真となることがない場合，φは真のままである」を含む．CTL を用いる場合，AG や EF などのように，経路演算子および状態演算子を組み合わせて，プロパティを作成する．

表 2-5　CTL

時相演算子	時相論理式	意味
経路演算子		
A (**a**ll)	**A** φ	現在状態から分岐する全ての経路で φ が真である.
E (**e**xists)	**E** φ	現在状態から分岐する経路のうち 少なくとも1つの経路で φ が真である.
状態演算子		
N (**n**ext) or **X** (ne**x**t)	**N** φ or **X** φ	φ は次の状態で真である.
G (**g**lobally)	**G** φ	φ はその後常に真である.
F (in the **f**uture)	**F** φ	φ はその後いずれかの時点で真となる.
U (**u**ntil)	ψ **U** φ	φ は現在または将来のある時点で真であり、 かつ，ψ はその時点まで真である. （その時点以降 ψ は真になるとは限らない）
W (**W**eak until)	ψ **W** φ	現在または将来のある時点でφが真ならば、 その直前までψは真である。 φ が真になることがない場合， ψ は真のままである.

※ φ，ψは命題

図 2-7 および図 2-8 に，CTL の概念を図示する．

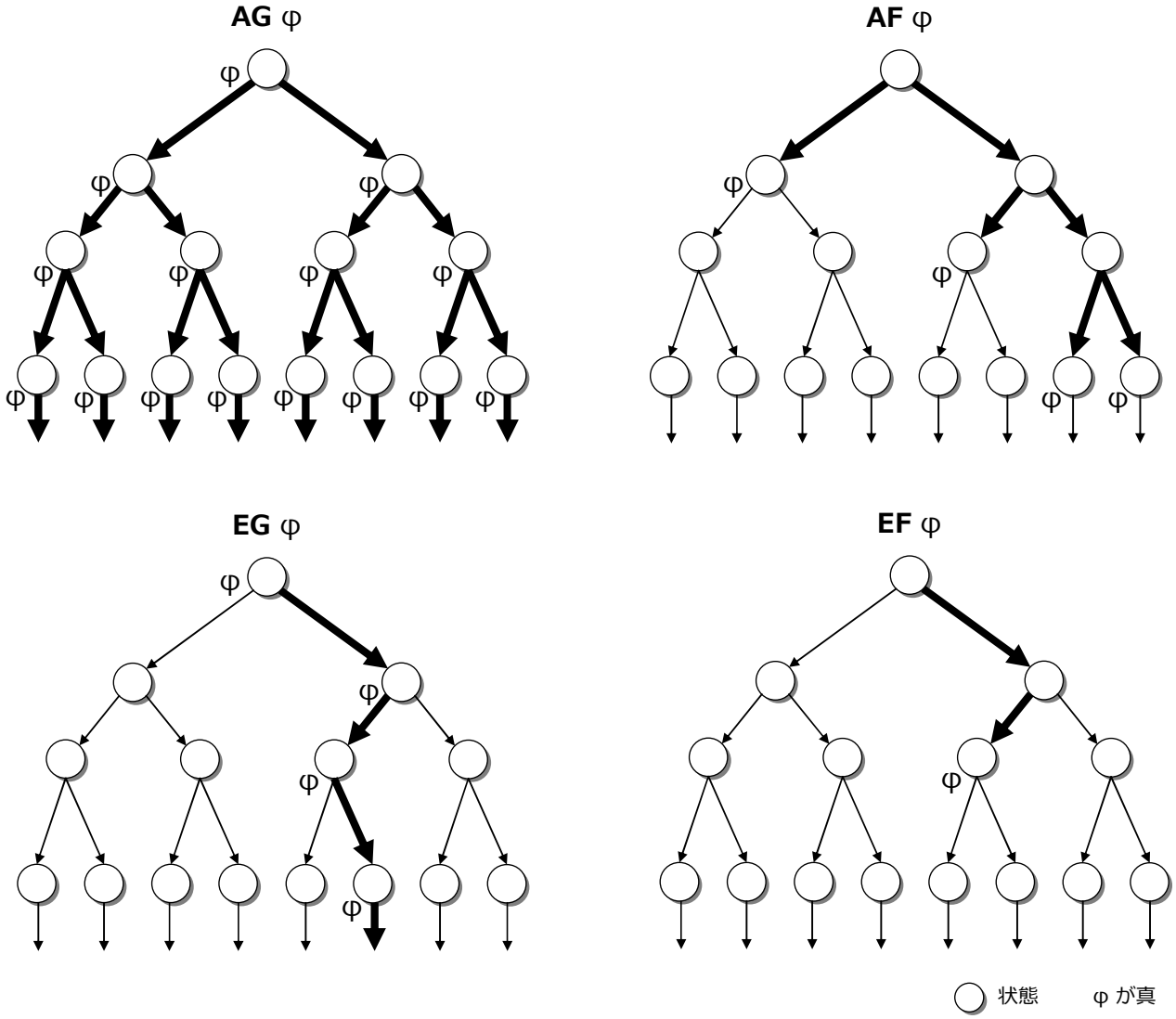


図 2-7　CTL の概念（その 1）

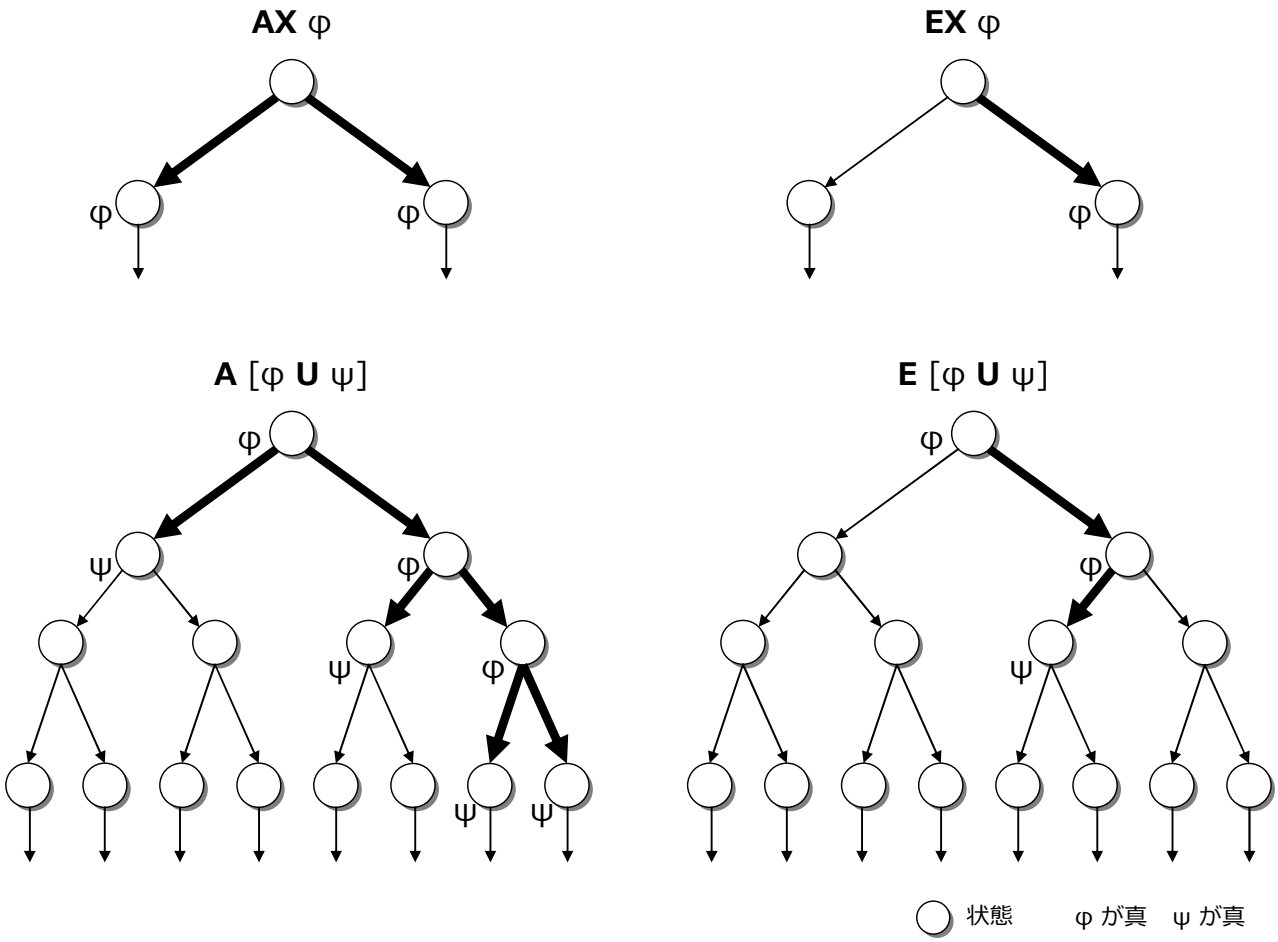


図 2-8　CTL の概念（その 2）

2.4　モデル検査ツール

モデル検査は，計算機上のモデル検査ツールに対して，状態遷移モデルおよびプロパティを与えることで実現される．表 2-6 に，産業界において適用実績が豊富なモデル検査ツールを示す．つぎより，表 2-6 に挙げた，それぞれのモデル検査ツールを詳述する．

表 2-6　モデル検査ツール

モデル検査ツール	開発元	状態遷移モデル	プロパティ	特徴
SPIN	Bell研究所	Promela	LTL	・非同期・並行システムを検証可能 ・Partial Order Reductionを用いたメモリの節約，高速化
SMV/NuSMV	Carnegie Mellon大学	SMVコード	・LTL（SMV） ・LTLおよびCTL（NuSMV）	・Symbolic Model Checking探索アルゴリズムを用いた状態空間の効率的な探索 ・電子回路の検証に適する ・非同期・並行システムを検証可能
UPPAAL	UPPSALA大学およびAALBORG大学	UPPAAL Graphicalモデル	Timed CTLのサブセット	・リアルタイム性を検証可能 ・GUIを用いたモデル化，反例の分析 ・非同期・並行システムを検証可能
BLAST	California大学Berkeley校	C言語ソースコードを基に自動生成	Assert関数を用いてソースコード中に記述	・ソースコードを直接入力可能 ・Lazy Abstractionによる状態遷移モデルの抽象化

2.4.1　SPIN

SPIN[27]は，Bell 研究所において開発された．Process Meta Language（以降，Promela という）[27]という専用言語を用いて，システムの状態遷移をモデル化する．プロパティの記述には，LTL を用いる．

SPIN を用いることで，非同期・並行システムを検証することができる．また，Partial Order Reduction[39]という，システムの性質に影響を与えないイベントの生起順序を固定する技術を用いて，計算機におけるメモリの節約と高速化を実現している．SPIN は文献や教科書が多く，代表的なモデル検査ツールである．

図 2-9 に，SPIN における状態遷移モデルのサンプルを示す．

```
bool turn, flag[2];   // the shared variables, booleans
byte ncrit;   // nr of procs in critical section

active [2] proctype user()   // two processes
{
  assert(_pid == 0 || _pid == 1);
again:
  flag[_pid] = 1;
  turn = _pid;
  (flag[1 - _pid] == 0 || turn == 1 - _pid);

  ncrit++;
  assert(ncrit == 1);   // critical section
  ncrit--;

  flag[_pid] = 0;
  goto again
}
```

図 2-9　SPIN における状態遷移モデル

2.4.2　SMV/NuSMV

SMV[28]は，Carnegie Mellon 大学において開発された．代入文や case 式からなる簡単な専用言語を用いて，システムの状態遷移をモデル化する．プロパティの記述には，CTL を用いる．SMV の後継ツールとして，NuSMV[29]がある．NuSMV では，プロパティの記述に LTL および CTL を用いることができる．

SMV および NuSMV は, 二分決定グラフを用いた Symbolic Model Checking[28]という探索アルゴリズムを用いて，状態空間の効率的な探索を実現している．SMV や NuSMV は，電子回路の検証に適しているといわれる．また，非同期・並行システムを検証することもできる．

図 2-10 に，NuSMV コードにおける状態遷移モデルのサンプルを示す．

```
MODULE main
VAR
  state1: {n1, t1, c1};
ASSIGN
  init(state1) := n1;
  next(state1) := case
                    (state1 = n1) & (state2 = t2): t1;
                    (state1 = n1) & (state2 = n2): t1;
                    (state1 = n1) & (state2 = c2): t1;
                    (state1 = t1) & (state2 = n2): c1;
                    (state1 = t1) & (state2 = t2) & (turn = 1):  c1;
                    (state1 = c1): n1;
                    TRUE : state1;
                  esac;

VAR
  state2: {n2, t2, c2};
ASSIGN
  init(state2) := n2;
  next(state2) := case
                    (state2 = n2) & (state1 = t1): t2;
                    (state2 = n2) & (state1 = n1): t2;
                    (state2 = n2) & (state1 = c1): t2;
                    (state2 = t2) & (state1 = n1): c2;
                    (state2 = t2) & (state1 = t1) & (turn = 2):  c2;
                    (state2 = c2): n2;
                    TRUE : state2;
                  esac;

VAR
  turn: {1, 2};
ASSIGN
  init(turn) := 1;
  next(turn) := case
                  (state1 = n1) & (state2 = t2): 2;
                  (state2 = n2) & (state1 = t1): 1;
                  TRUE : turn;
                esac;
```

図 2-10　NuSMV における状態遷移モデル

2.4.3 UPPAAL

UPPAAL[30]は，UPPSALA 大学および AALBORG 大学の共同研究により開発された．UPPAAL における状態遷移モデルの作成は，Graphical User Interface（以降，GUI という）を用いて，直感的に行うことができる．プロパティの記述には，Timed CTL[41]のサブセットを用いる．

UPPAAL の特徴は，システムのリアルタイム性（イベントに対し，一定時間内に処理が完了する性質）を評価することができる点である．UPPAAL の状態遷移モデルは，時間オートマトン[40]に対応している．また，状態遷移のモデル化および反例の分析を，GUI を用いて直感的に行うことができる．図 2-11 に，UPPAAL における状態遷移モデルのサンプルを示す．図 2-11 において，◎は初期状態，○はその他の状態を表す．矢印は状態遷移，矢印に付記される条件式（"=="）は遷移条件を表す．矢印に付記される代入式（"="）は，モデル上の変数に対する値の設定を表す．図 2-12 に，UPPAAL において反例が出力された場合のサンプルを示す．図 2-12 の上部に示すウィンドウには，プロパティが成り立たない状態に至るまでの状態遷移のパスが出力される．図 2-12 上部のウィンドウにおいて，状態遷移のパスを遡る場合，それに対応する状態遷移が，図 2-12 の下部に示す状態遷移モデルに表示される．また，UPPAAL を用いて，非同期・並行システムを検証することができる．

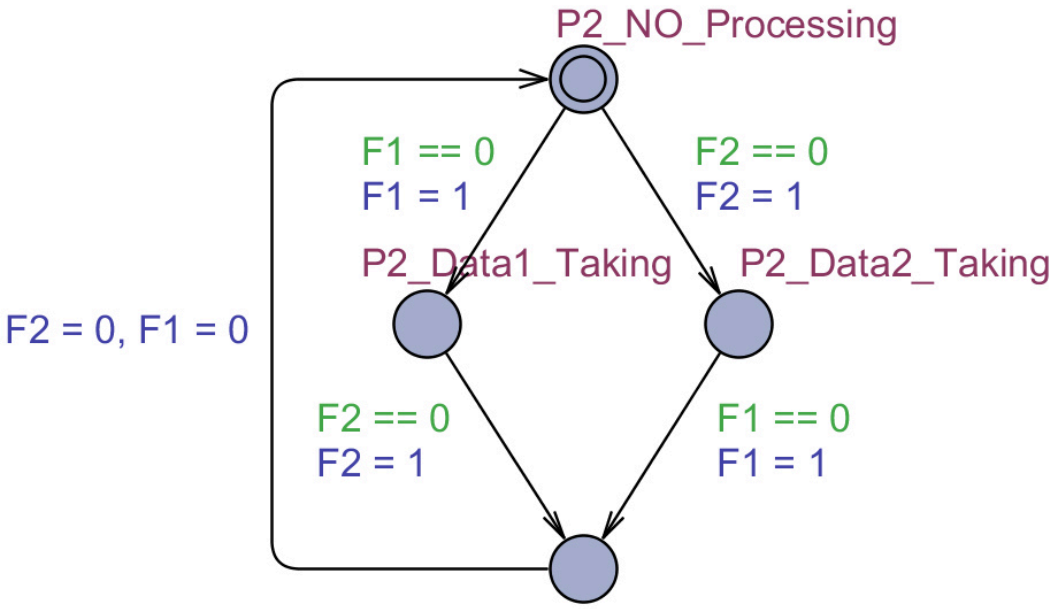


図 2-11　UPPAAL における状態遷移モデル

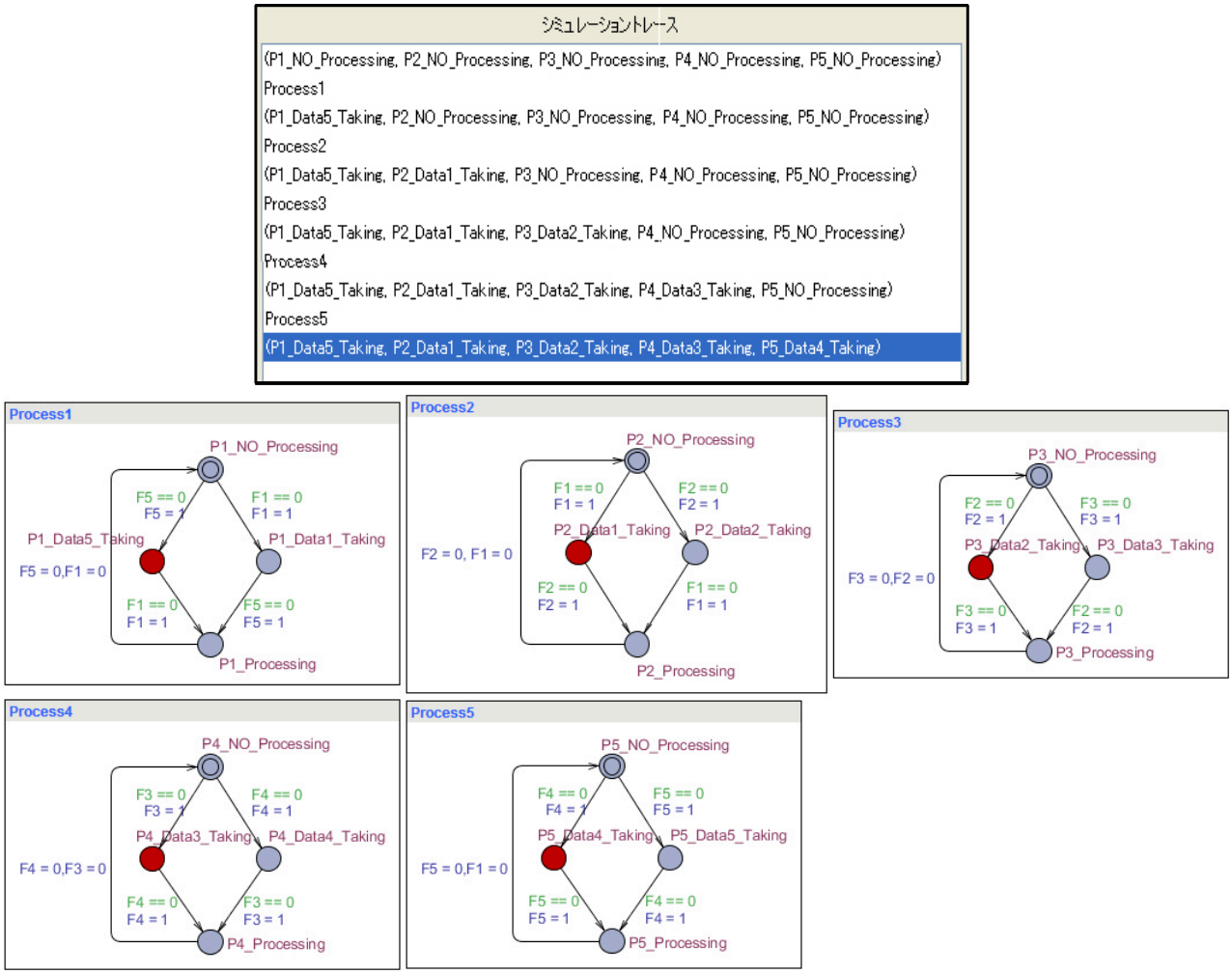


図 2-12　UPPAAL における反例の出力

2.4.4 BLAST

BLAST[42]は，California 大学 Berkeley 校において開発された．BLAST では，C 言語で記述されたソースコードを入力として, ソースコードから状態遷移モデルを自動生成する．プロパティの記述は，満たすべき条件を引数として与えた assert 関数を，ソースコード中に挿入することで実現される．

BLAST の特徴は，C 言語で記述されたソースコードを直接，ツールに入力することができる点である．BLAST では，Lazy Abstraction[43]という，状態遷移モデルの抽象化，検証，詳細化を繰り返す方法で，状態遷移モデルを自動生成する．このようなモデル検査を，従来型のモデル検査（Classic Model Checking）に対して，モダン・モデル検査（Modern Model Checking）[31]という．従来型のモデル検査では，検証者が検証対象システムを抽象化することで，適切な規模の状態遷移モデルを得る．一方，モダン・モデル検査では，抽象化技法を用いて，ソースコードやデザイン記述から，モデル検査を実行可能な状態遷移モデルを自動生成する．

モダン・モデル検査に属するモデル検査ツールとして，BLAST の他に，C 言語のソースコードを入力とする SLAM[44], Java 言語のソースコードを入力とする JPF[45]や Bandera[46]などがある．

2.5 複雑なシステムへのモデル検査適用時の課題

産業界におけるモデル検査の適用を通して, モデル検査のそれぞれの工程における課題が識別されている[47][48][49]．図 2-13 に，システムに対してモデル検査を適用する際の，モデル検査適用者の視点から見た課題を示す．モデル検査の工程 1 には，以下の課題がある．

1. 仕様書などから，検証対象システムの状態遷移を識別する困難さ．
   仕様書に状態遷移図が記述されていない場合がある．また，状態遷移図が記述されていたとしても，状態遷移図には記述されていない状態遷移が存在する場合がある．
2. モデル検査ツール独自の状態遷移モデルの表現形式を習得する必要性．
   状態遷移モデルを作成する言語，方法が，モデル検査ツールによって異なる．

工程 2 には，以下の課題がある．

3. 安全性プロパティを導出する困難さ
   活性プロパティは，検証対象の上位の仕様書に定義されている要件や機能である場合が多い．しかし，安全性プロパティは，仕様書において明には記述されていない．
4. 時相論理を用いてプロパティを記述する困難さ．
   LTL や CTL などを用いて検査式を表現するには，数学的な知識が必要である．

工程 3 には，以下の課題がある．

5. モデル検査ができない，終了しない（状態爆発の発生）
   システムの状態数によっては，計算機のメモリ資源が枯渇し，モデルが検証できない，もしくは，モデル検証が現実的な時間では終了しない現象（状態爆発）が発生することがある．

工程 4 には，以下の課題がある．

6. 反例解析の手間
   モデル検査ツールから出力される反例は，可読性が低く，その量も膨大である．反例の中から，状態遷移モデルの基になった仕様に不具合があるか否かを分析するには，ノウハウが必要である．

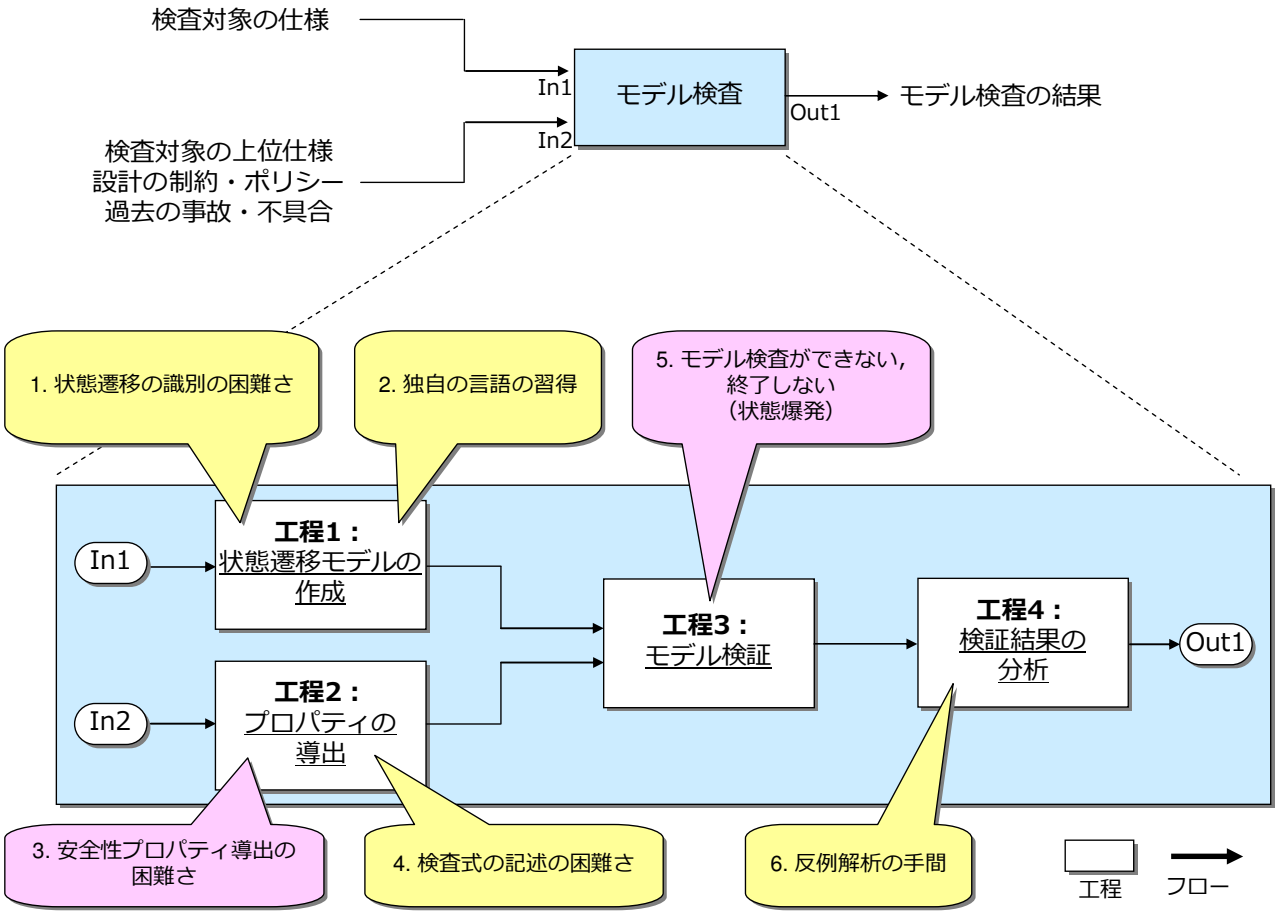


図 2-13　モデル検査適用時の課題

これらのモデル検査の課題に対して, 多くの取り組みが行われている. しかし, 図 2-13 の課題 3 および課題 5 については，システムが複雑になるつれ，その解決が困難になる.

図 2-13 の課題 1 については，モデル検査の敷居を下げることを目的として，近年，モデル検査に関する良質の参考文献が発表されている[50][51][52]．それらの中には，例題を中心に，状態遷移の識別方法が解説されている文献も多く存在する．課題 2 については，Unified Modeling Language（以降，UML という）[53]で記述された仕様を基に状態遷移モデルを自動で作成し，モデル検査を実施する取り組みが行われている[54][55][56]．UML は，ソフトウェア開発において広く浸透している開発技術である．UML のダイアグラムから，モデル検査ツールに対応した状態遷移モデルが自動生成される．それにより，モデル検査ツール独自の状態遷移モデルの表現形式を習得する労力が軽減される．課題 4 については，非形式的に表現されるプロパティを基に，時相論理による正確なプロパティを作成する Spec Pattern[57]が提案されている．Spec Pattern ではプロパティを，スコープとパターンの 2 つの観点で捉える．そして，パターンおよびスコープと, それらに対応する時相論理のマッピングを提供している. Spec Pattern を用いることで, 時相論理体系に精通していない技術者であっても, LTL や CTL による正確なプロパティの作成が期待できる．課題 6 については，膨大に出力される反例をフィルタリングすることで，真に分析すべき結果に着目する取り組みが行われている[48]．また，UPPAAL のように，GUI を用いた反例分析を可能にするツールも出現している.

しかし，図 2-13 の課題 3 については，先行研究が存在するものの，プロパティ導出のルールが明確ではなく，課題の解決には至っていない．課題 5 については，状態空間を表現するデータ構造，状態空間の探索アルゴリズムの工夫，状態遷移モデルの抽象化により，状態爆発を回避する方法が提案され，モデル検査ツールに実装されている[31][49]．それらの方法により，状態爆発の回避に成果を上げている．しかし，検証対象のシステムが複雑になるにつれ，それらの方法を用いても状態爆発が発生する可能性が高くなる．また，分割統治法（Divide and Conquer）の考えに基づき，検証対象のシステムを分割してモデル検査する Compositional Verification[58][59]により，状態爆発を回避する方法が提案されている．しかし，Compositional Verification に関する先行研究は，検証対象のシステムが複雑になるにつれ，その適用が困難になる場合がある.

本論文では，複雑なシステムへのモデル検査適用時に問題となる図 2-13 の課題 3（図 1-4 に示す課題 2 に対応）および課題 5（図 1-3 に示す課題 1 に対応）に焦点を当て，それらの解決を図る．それにより，産業界におけるモデル検査の適用を加速させ，社会に浸透する複雑な組込みシステムの信頼性を向上させる．

第3章　構成要素および構成要素の連携部分に着目したモデル検査

本章では，トレーサビリティマトリクスを用いて，システムにおける構成要素の連携部分を抽出し，構成要素および構成要素間の連携部分をモデル検査する手法を述べる．本章で述べる手法は，第 1 章で述べた課題 1 を解決する．

3.1　はじめに

複雑なシステムは，それを構成する複数の要素で構築される．また，それらの構成要素が連携しながら，システムの機能を実現する．例えば，3.4 で述べる不定形剛体運搬ロボットシステムの場合，測定サブシステムの測定結果を基に，ロボット制御サブサブシステムが周辺状況を判断し，ロボットアームおよびロボットハンドを動作させることなどである．より詳しくは，システムの機能を実現するために，システムの構成要素における処理がインタフェースを介し，他の構成要素の処理と連携することである．複雑なシステムの正しさを検証する場合，システムの構成要素を組み合わせた検証が重要である．

複雑なシステムの構成要素のすべてを対象としてモデル検査を実施する場合，状態爆発が発生するという課題がある．図 3-1 に，システムの構成要素を一括してモデル検査する場合の課題を示す．一般に，モデル検査の対象が単純なシステムの場合，構成要素の変数は少なく，構成要素間の連携も少ない．システムの状態数が少ないため，状態爆発が発生する可能性は低い．しかし，システムが複雑になるにつれ，構成要素の変数および構成要素間の連携は増加する．その結果，システムの状態数が膨大になり，モデル検査が現実的な時間で終了しない可能性が高くなる．

本論文では, システムの構成要素および構成要素間の連携部分をそれぞれ個別にモデル検査する手法を提案する．連携に関連する仕様は，アーキテクチャ設計プロセスで作成するトレーサビリティマトリクスを用いて，構成要素の仕様書および構成要素間のインタフェースの仕様書から漏れなく抽出する．アーキテクチャ設計とは，システムの構成要素に対して，システムに要求される機能・性能を配分し，構成要素の仕様および構成要素間のインタフェース仕様を明確化する技術である．アーキテクチャ設計のプロセスは，IEEE1220 などのシステムズエンジニアリング標準[7][8][9]に規定されている．すべての構成要素を一括してモデル検査するのではなく，構成要素および構成要素の連携部分に着目し，

それぞれを個別にモデル検査する．それにより，状態爆発を回避する．

提案手法の有効性を確認するために，不定形剛体運搬システムの連携部分を適用対象とする，提案手法の評価実験を行った．実験の結果，状態爆発を発生させることなく，構成要素を組み合わせなければ発見できない不具合を検出することができた．

本章の構成はつぎの通りである．3.2 において，関連研究を述べる．3.3 において，提案手法を詳述する．3.4 において，提案手法に対する評価実験の結果を示す．3.5 において，提案手法の有効性を考察する．3.6 において，本論文をまとめる．

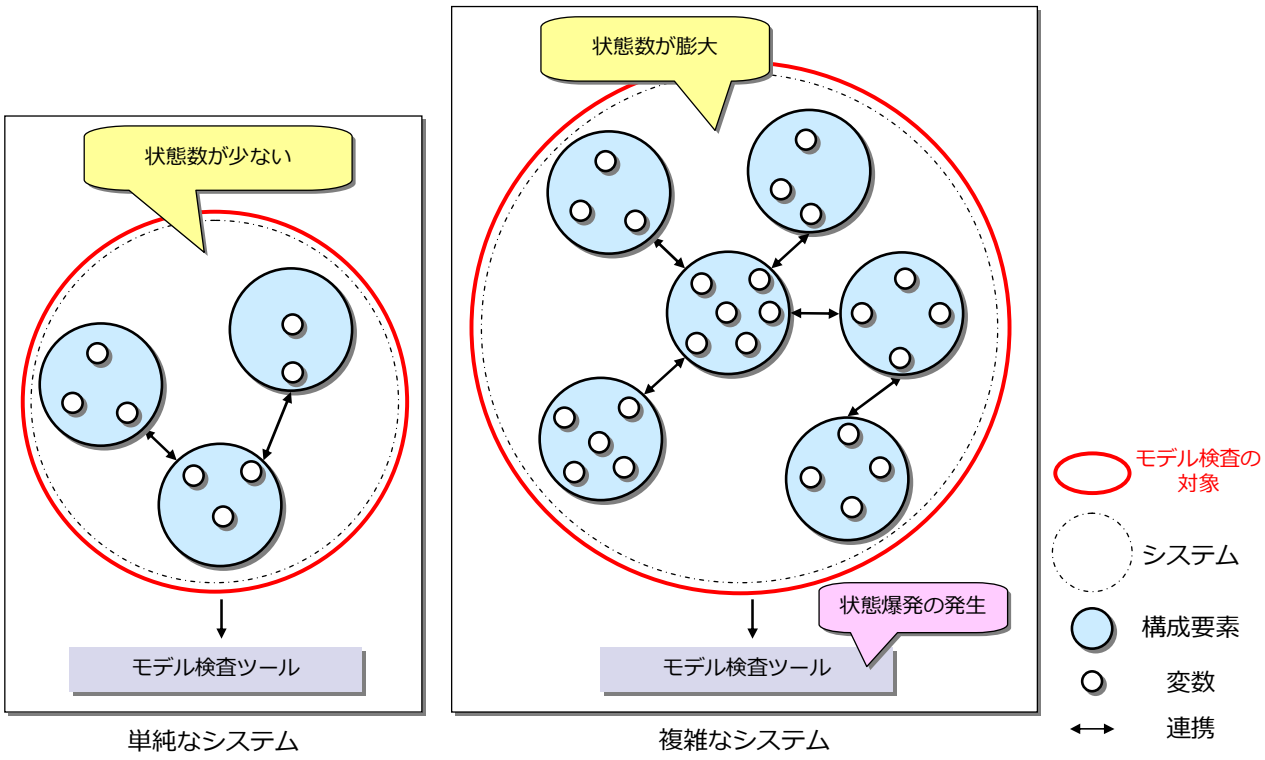


図 3-1　構成要素を一括してモデル検査する場合の課題

3.2 関連研究

複雑なシステムに対してモデル検査を適用する際の, 状態爆発を回避する方法として，Compositional Verification[58][59] が提案されている．Compositional Verification とは，モデル検査の対象となるシステムをモジュールに分解し，それぞれのモジュールを検証することで，システム全体を検証する方法である．Compositional Verification の具体的な方法として，Compositional Minimization[60] および Assume-Guarantee Reasoning[61][62]が存在する．

Compositional Minimization について，Clarke らは，[60]において，検証対象である構成要素と，それ以外の構成要素に着目する方法を提案している．図 3-2 に，Clarke らによる Compositional Minimization を示す．[60]では，検証対象以外の構成要素を抽象化し，検証対象に対する Interface process を作成する．検証対象である構成要素と，Interface Process を組み合わせてモデル検査する．それをすべての構成要素に対して実施することで，システム全体をモデル検査する．図 3-2 の例の場合，構成要素 A に対して構成要素 B および構成要素 C を抽象化し, Interface Process BC’ を作成する. そして, 構成要素 A および Interface Process BC’ を組み合わせてモデル検査する. 構成要素 B および構成要素 C に対しても，同様の方法でモデル検査する．

この方法により，システム全体をモデル検査する場合と比較して，状態遷移モデルを小さくすることができる．また，Compositional Minimization の方法が形式的に定義されている．しかし，[60]では，構成要素と Interface Process を組み合わせてモデル検査する. そのため, Interface Process が小さくならないシステム，つまり，構成要素およびその連携が多いシステムには不向きである．Interface Process が小さくならない場合には，モデル検査の対象となる状態遷移モデルが大きくなる．その場合，比較的多くの計算機リソースが必要となる．また，Interface Process の作成方法が明らかにされていない．

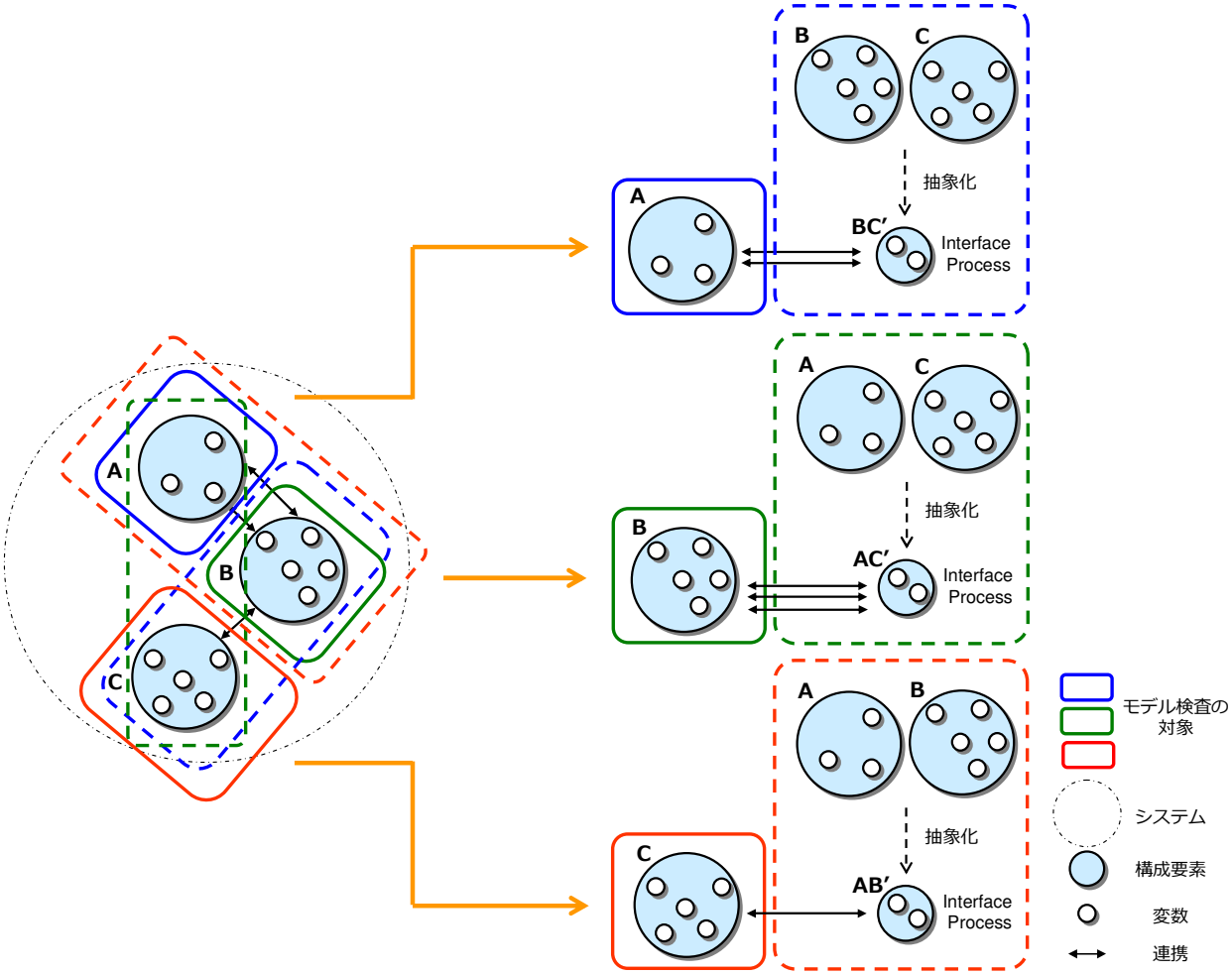


図 3-2　Compositional Minimization

Assume-Guarantee Reasoning について，Grumberg らは，[61]において，構成要素の検証の際に構成要素の環境に関する Assumption を設定し，構成要素それぞれをモデル検査する方法を提案している．図 3-3 に，Grumberg らによる Assume-Guarantee Reasoning を示す．[61]では，検証対象である構成要素の連携動作などの環境を，Assumption として識別する．識別した Assumption の下で，検証対象である構成要素をモデル検査する．そして，他の構成要素が，検証対象である構成要素の Assumption を満たすことをモデル検査で確認する．それをすべての構成要素に対して実施することで，システム全体をモデル検査する．図 3-3 の例の場合，構成要素 A の連携動作などの環境を Assumption P とし，P の下で構成要素 A をモデル検査する．そして，構成要素 B および構成要素 C が P を満たすことをモデル検査で確認する．構成要素 B および構成要素 C についても，同様の方法でモデル検査する．

この方法により，構成要素の単位でのモデル検査が可能になる．そのため，モデル検査の対象となる状態遷移モデルが小さくなり，少ない計算機リソースでの検証が可能になる．また，Assume-Guarantee Reasoning の方法が形式的に定義されている．しかし，[61]では，検証対象のモデル検査における Assumption を手動で生成する必要がある．検証対象となる構成要素の Assumption の生成は，他のすべての構成要素が Assumption を満たすことを考慮して行う必要がある．そのため，検証対象のシステムやモデル検査に関する高度な知識が必要となる．また，[61]では，検証対象以外の構成要素において，検証対象の Assumption に対するモデル検査を実施する必要がある．そのため，構成要素が多いシステムの場合，構成要素に対するモデル検査の回数が増大する．

[61]の Assumption 生成に関する課題を解決するために，Cobleigh らは，[62]において，Assumption を自動で生成する Assume-Guarantee Reasoning を提案している．[62]では，検証対象および他の構成要素に対するモデル検査時の反例を利用することで，検証対象の Assumption を自動で生成している．それにより，実施者が Assume-Guarantee Reasoning を適用する際の負荷を低減することができる．しかし，[61]と同様に，システムの構成要素が多い場合，構成要素に対するモデル検査の回数が増大する．

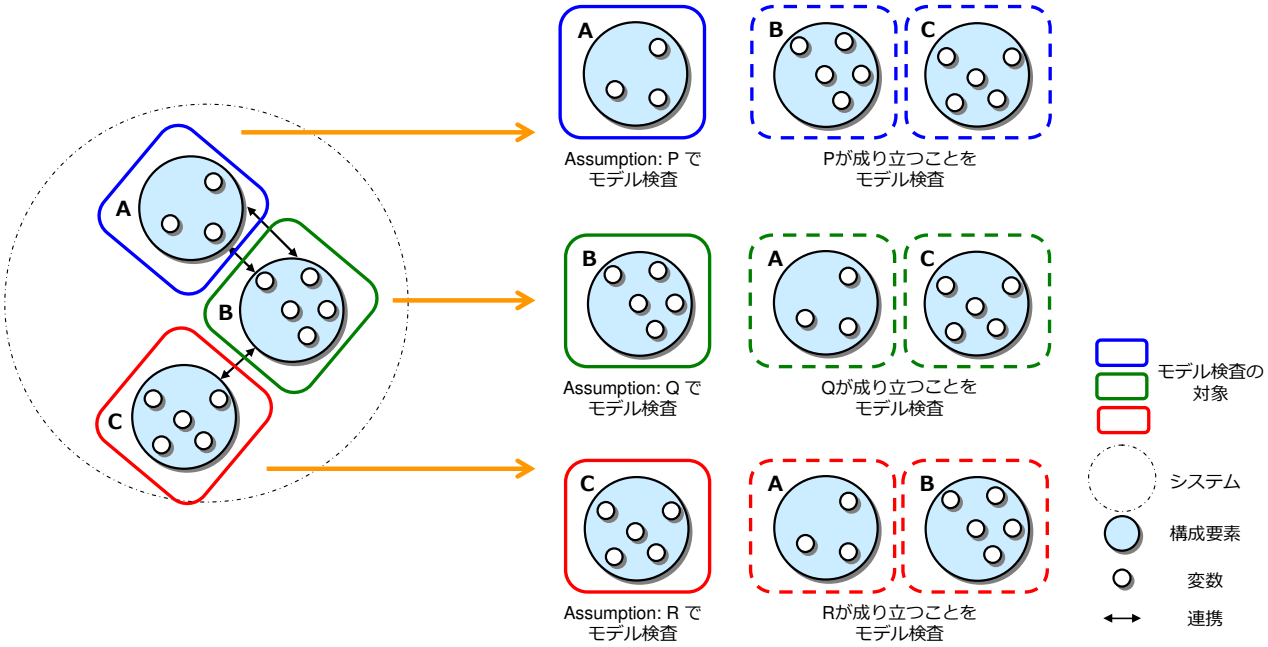


図 3-3　Assume-Guarantee Reasoning

一方，提案手法では，構成要素および構成要素の連携部分を，それぞれ個別にモデル検査することで，それらの状態遷移モデルを小さくすることを図る．それにより，構成要素および構成要素の連携が多いシステムへのモデル検査の適用を可能にする．また，構成要素の連携部分を識別する際に，構成要素の機能間の関連性を端的に表現するトレーサビリティマトリクスを用いる．それにより，システムの機能を実現するために連携する，それぞれの構成要素の機能を漏れなく容易に識別することができる．

表 3-1 に, Compositional Verification に関する先行研究と提案手法の比較結果をまとめる．

表 3-1　Compositional Verification の先行研究と提案手法の比較

	Compositional Minimization [※1]	Assume-Guarantee Reasoning [※2]	Assume-Guarantee Reasoning with automatic generation of Assumption [※3]	提案手法
適用性	× 構成要素， 構成要素の連携が多い システムには不向き	× 構成要素の多い システムには不向き	× 構成要素の多い システムには不向き	○ 構成要素， 構成要素の連携が多い システムに適用可能
資源性	△ Interface Processが 小さくならない場合， 状態遷移モデルが大	○ 構成要素の単位で検証するため， 状態遷移モデルが小	○ 構成要素の単位で検証するため， 状態遷移モデルが小	○ 構成要素、 構成要素の連携毎に 検証するため， 状態遷移モデルが小
容易性	× Interface Processの 作成方法が不明確	× Assumptionを手動で生成	○ Assumptionを自動で生成	○ 理解性の高い トレーサビリティマトリクスで 構成要素の連携部分を抽出
形式性	○ 手法を形式的に定義	○ 手法を形式的に定義	○ 手法を形式的に定義	× 手法を非形式的に定義

※1 Clarke, E. M., Long, D. E. and McMillan, K. L.: Compositional model checking, Proceedings of the Fourth Annual Symposium on Logic in Computer Science, pp.353-362 (1989).
※2 Grumberg, O. and Long, D.: Model checking and modular verification, ACM Transactions on Programming Languages and Systems, Vol.16, No.3, pp.843-871 (1994).
※3 Cobleigh, J.M., Giannakopoulou, D. and Pasareanu, C. S.: Learning assumptions for compositional verification, Proceeding of 9th TACAS, LNCS Vol.2619, pp.331-346 (2003).

3.3 提案手法

複雑なシステムに対するモデル検査は, 構成要素および構成要素間の連携部分を，それぞれ個別にモデル検査することで実現される．提案手法を図 3-4 に示す.

提案手法は，構成要素を対象としたモデル検査，構成要素間の連携部分を対象としたモデル検査で構成される．まず，それぞれの構成要素の仕様を基に，構成要素の状態遷移モデルを作成する．そして，構成要素の仕様に対応する上位仕様（システムの仕様）や，第 4 章に述べる課題 2 に対する提案手法から，構成要素のプロパティを導出する．構成要素の状態遷移モデルおよびプロパティを用いて，それぞれの構成要素に対するモデル検査を実施する．つぎに，システム開発のアーキテクチャ設計時に作成されるトレーサビリティマトリクスを用いて，それぞれの構成要素の仕様および構成要素間のインタフェース仕様から，構成要素間の連携部分を抽出する．抽出した構成要素の連携部分の仕様を基に，連携部分の状態遷移モデルを作成する．そして，連携部分に対応する上位仕様（システムの仕様）や，第 4 章に述べる課題 2 に対する提案手法から，連携部分のプロパティを導出する．構成要素の連携部分の状態遷移モデルおよびプロパティを用いて，連携部分に対するモデル検査を実施する．構成要素および構成要素の連携部分を個別にモデル検査することで，状態爆発を回避しながら，システム全体を検証する.

3.3.1 において，提案手法のプロセスを述べる．3.3.2 において，トレーサビリティマトリクスを用いた連携部分の抽出を述べる.

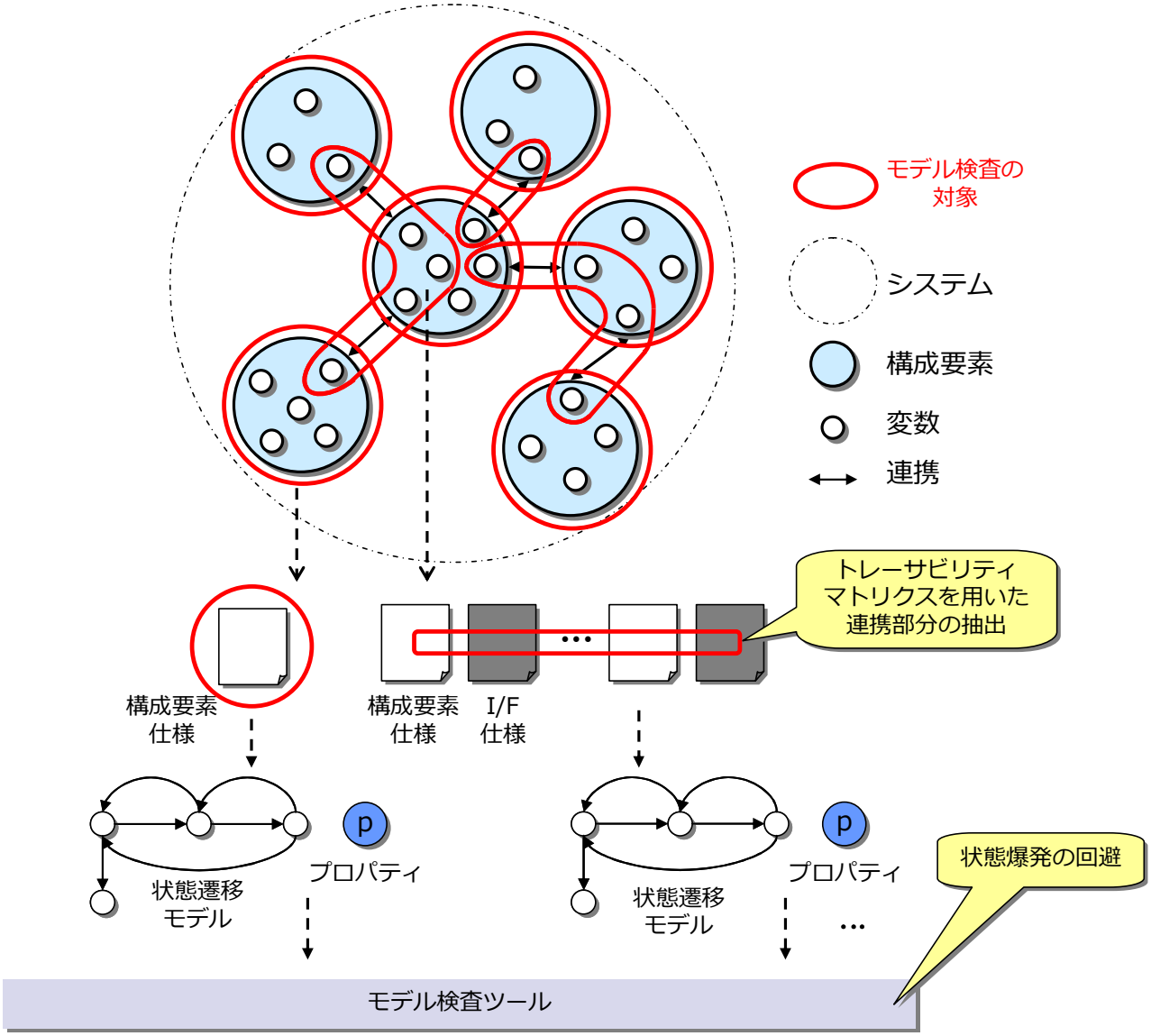

モデル検査の
対象
システム
構成要素
変数
連携
トレーサビリティ
マトリクスを用いた
連携部分の抽出
構成要素
仕様
構成要素
仕様
I/F
仕様
p
プロパティ
状態遷移
モデル
p
プロパティ
状態遷移
モデル
状態爆発の回避
モデル検査ツール

図 3-4　提案手法

3.3.1 提案手法のプロセス

図 3-5 に，提案手法のプロセスを示す．提案手法に対して，システム開発におけるつぎの成果物が与えられる．

- システムの仕様書
- システムにおける構成要素の仕様書
- 構成要素間のインタフェース仕様書
- トレーサビリティマトリクス

これらを基に，提案手法では，構成要素および構成要素の連携部分に着目したモデル検査を実施する．以下の記述における番号（1.から 9.）は，図 3-5 のブロック内に示される各番号に対応する．図 3-5 の「システムの構成要素のループ」の n は，システムの構成要素の総数を表す．1.から 4.の処理は，システムのそれぞれの構成要素に対して実施される．「構成要素の連携部分のループ」の m は，トレーサビリティマトリクスを用いて抽出した，構成要素の連携部分の総数を表す．6.から 9.の処理は，構成要素の連携部分のそれぞれに対して実施される．

1. 構成要素 i の仕様を基に，構成要素 i に対する状態遷移モデル i を作成する．
2. 構成要素 i に関するプロパティ i を導出する．活性プロパティについては，構成要素 i の上位仕様であるシステム仕様などから導出する．安全性プロパティについては，本論文の第 4 章に示す，課題 2 に対する提案手法を用いて導出する．
3. 状態遷移モデル i およびプロパティ i を用いて，構成要素 i に対するモデル検証を実施する．
4. モデル検査ツールから出力される反例を基に，状態遷移モデル i およびプロパティ i によるモデル検証の結果を分析する．
5. トレーサビリティマトリクスを用いて，それぞれの構成要素の仕様および構成要素間のインタフェース仕様から，構成要素間の連携部分 j を抽出する．トレーサビリティマトリクスを用いた連携部分の抽出については，3.3.2 に詳述する．
6. 抽出した連携部分 j の仕様を基に，連携部分 j に対する状態遷移モデル j を作成する．
7. 連携部分 j に関するプロパティ j を導出する．活性プロパティについては，

連携部分 j の上位仕様であるシステム仕様などから導出する．安全性プロパティについては，本論文の第 4 章に示す，課題 2 に対する提案手法を用いて導出する．

8. 状態遷移モデル j およびプロパティ j を用いて, 連携部分に関するモデル検証を実施する．
9. モデル検査ツールから出力される反例を基に，状態遷移モデル j およびプロパティ j によるモデル検証の結果を分析する．

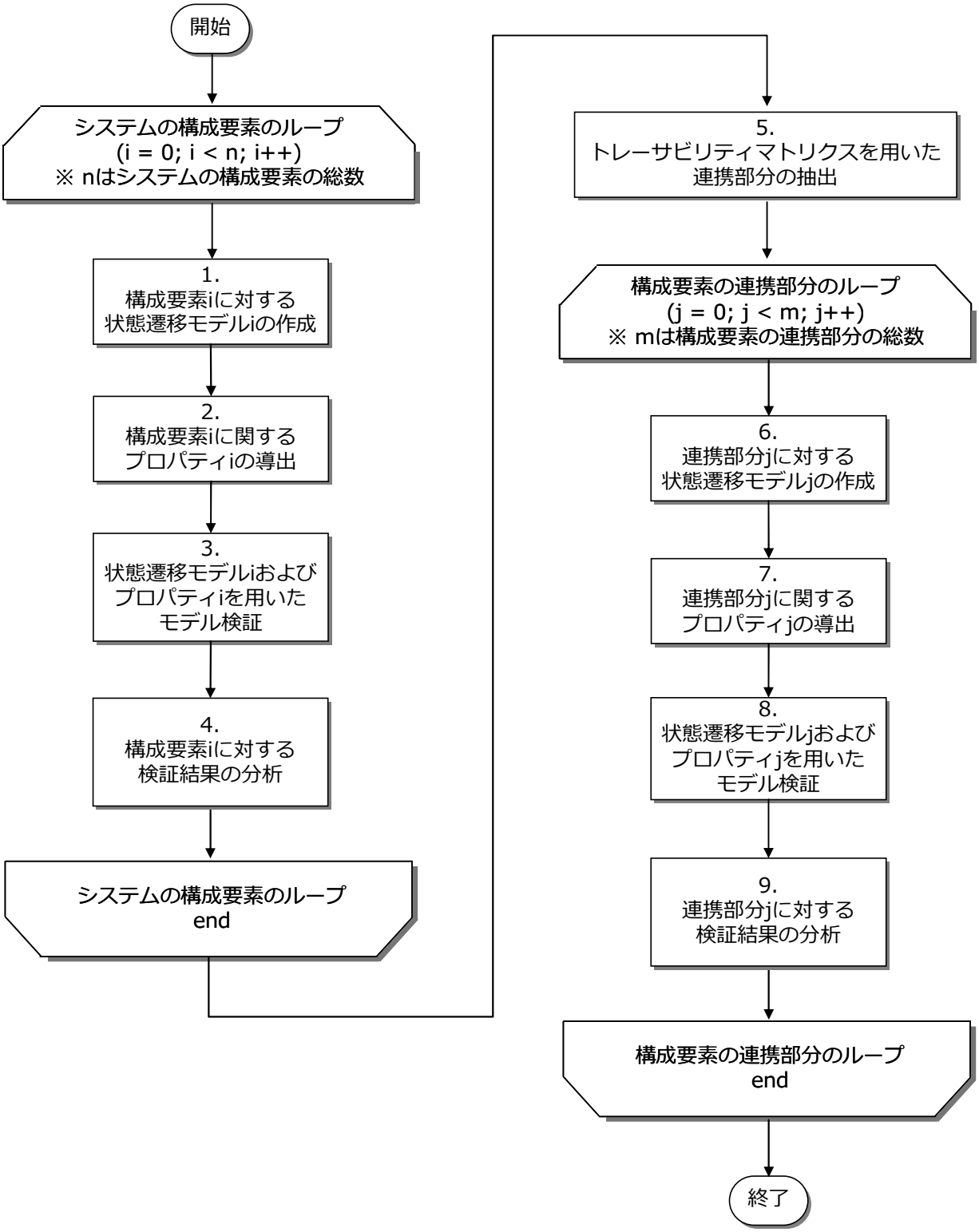


図 3-5　構成要素および構成要素の連携部分に着目したモデル検査のプロセス

3.3.2　トレーサビリティマトリクスを用いた連携部分の抽出

図 3-6 に, アーキテクチャ設計時に作成されるトレーサビリティマトリクスを用いた連携部分の抽出方法を示す．トレーサビリティマトリクスとは，システム仕様と，それを分解・詳細化した構成要素の仕様との対応関係がまとめられるマトリクスである．図 3-6 の例の場合，システム仕様における「4.1 X 機能」は，つぎに示す下位の仕様に分解・詳細化されている．トレーサビリティマトリクスでは，それらの対応関係が図 3-6 の下部に示すようにまとめられる．

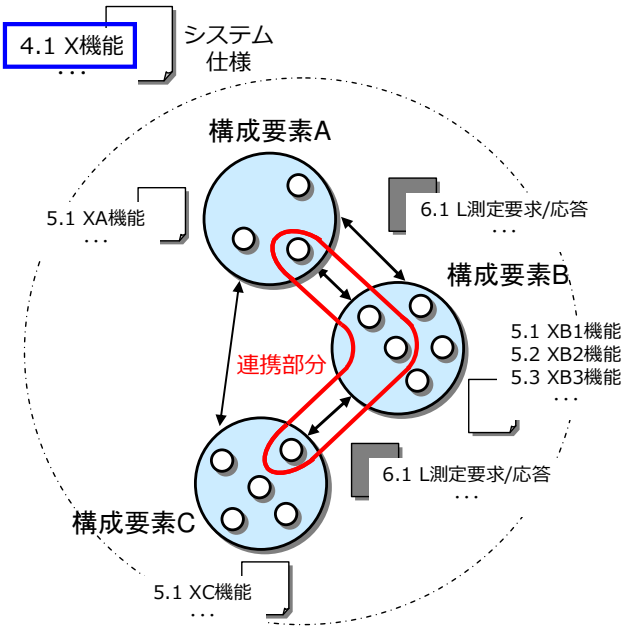
- 構成要素 A 仕様：「5.1 XA 機能」
- 構成要素 B 仕様：「5.1 XB1 機能」,「5.2 XB2 機能」,「5.3 XB3 機能」
- 構成要素 C 仕様：「5.1 XC 機能」
- 構成要素 A および構成要素 B 間インタフェース仕様：「6.1 L 測定要求/応答」
- 構成要素 B および構成要素 C 間インタフェース仕様：「6.1 L 測定要求/応答」

トレーサビリティマトリクスを用いることによって, 構成要素の連携部分を抜けなく容易に把握することができる．トレーサビリティマトリクスには，システム仕様と，それの下位にあたる仕様との対応関係がまとめられている．言い換えると，トレーサビリティマトリクスにまとめられる，システム仕様の一機能に対応する下位仕様の複数の機能およびインタフェースは，それら複数の機能がインタフェースを介し連携することで，システム仕様の一機能を実現することを意味する．図 3-6 の例の場合，前述の「5.1 XA 機能」,「5.1 XB1 機能」,「5.2 XB2 機能」,「5.3 XB3 機能」,「5.1 XC 機能」, 構成要素 A および構成要素 B 間インタフェース仕様の「6.1 L 測定要求/応答」, 構成要素 B および構成要素 C 間インタフェース仕様の「6.1 L 測定要求/応答」は, それぞれが連携することで, システム仕様の「4.1 X 機能」を実現する．

抽出した連携部分は，さらにそれらが連携の関係を有する可能性がある．トレーサビリティマトリクスを用いて抽出した，それぞれの連携部分は，それぞれの連携部分同士で，システムの資源を共有している場合がある．そのような共有関係を有する，それぞれの連携部分は，複合的な連携部分と捉えることができる．したがって，トレーサビリティマトリクスによる連携部分の抽出後，抽出した仕様の中から資源や変数を識別する．それぞれの連携部分で抽出した資

源や変数を比較する．連携部分同士が資源や変数の共有関係を有する場合には，共有関係を有するそれぞれの連携部分を統合し，1 つの連携部分とする．

また，トレーサビリティマトリクスを用いることによって，連携部分の活性プロパティを容易に識別することができる．連携部分は，それに対応するシステム仕様の機能を実現する必要がある．すなわち，連携部分に対応するシステムの仕様は，連携部分の活性プロパティに相当する．



トレーサビリティマトリクス

システム仕様	構成要素A仕様	構成要素B様	構成要素C仕様	構成要素A/ 構成要素B I/F仕様	構成要素B/ 構成要素C I/F仕様
4.1 X機能	5.1 XA機能	5.1 XB1機能 5.1 XB2機能 5.1 XB3機能	5.1 XC機能	6.1 L測定要求/応答	6.1 L測定要求/応答
… 連携部分の 活性プロパティ …		…	…	…	… 連携部分

図 3-6　トレーサビリティマトリクスを用いた連携部分の抽出

3.4 評価実験

提案手法の評価実験を行い，提案手法の有効性を確認する．提案手法は，構成要素に対する検証および構成要素の連携部分に対する検証に大別される．評価実験においては，構成要素の連携部分の検証に着目し，産業界の実例である複雑なシステムに対して提案手法を適用する．適用を通して，状態爆発を発生させることなく，構成要素の連携の正しさが検証できることを確認する．

3.4.1 適用事例

評価実験の事例として，不定形剛体運搬ロボットシステムを選定した．図 3-7 に，不定形剛体運搬ロボットシステムを示す．不定形剛体運搬ロボットシステムを選定した理由は，本システムが産業界の実例であり，かつ，複雑なシステムだからである．不定形剛体運搬ロボットシステムとは，形状・寸法が一定ではない重量のある剛体の把持，運搬，据置を行う産業用ロボットである．不定形剛体運搬ロボットシステムの特徴は，不定形剛体の把持および据置作業に対して，つぎに示す強い自律性が求められる点である．不定形剛体運搬ロボットシステムにおいて，不定形剛体を把持する領域は限定されているものの，剛体の形状・寸法，剛体を把持する位置，剛体の姿勢は剛体ごとに変化する．システムは，剛体を把持するにあたり，剛体の形状・寸法，位置，姿勢を正確に知る必要がある．また，不定形剛体を据え置く領域は限定されているものの，その領域の中で剛体を据え置く位置が変化する．システムは剛体を据え置くにあたり，他の剛体が存在しない位置もしくは剛体が敷き詰められた領域において，最も標高が低い位置を正確に知る必要がある．

図 3-8 に，不定形剛体運搬ロボットシステムのシステム構成を示す．不定形剛体運搬ロボットシステムは，ロボット制御サブシステムおよび形状測定サブシステムで構成される．ロボット制御サブシステムは，つぎに示すコンポーネントで構成される．

- ロボットハンド
- ロボットアーム
- ロボット動作のプログラミングや緊急停止を行うティーチペンダント
- ロボットハンドを制御するコントローラ

- ロボットアームおよびティーチペンダントを制御するコントローラ
- 運用者が作業指示やシステムのステータス確認を行うコンソール
- ロボット制御サブシステムの統合制御を行うロボット制御計算機

ロボット制御計算機は，つぎに述べる形状測定サブシステムとのインタフェースを有する．

形状測定サブシステムは，つぎに示すコンポーネントで構成される．

- 3 次元形状を測定するレーザスキャナ
- レーザスキャナ上下動機構
- レーザスキャナおよびレーザスキャナ上下動機構を制御する測定制御計算機

測定制御計算機はロボット制御サブシステムとのインタフェースを有する．

形状測定サブシステムは，剛体を把持する際に，剛体の形状・寸法，位置，姿勢を測定する．剛体を据え置く際に，据置領域の凹凸状況を測定する．形状測定サブシステムは，ロボット制御サブシステムに対して，測定した情報を送信する．ロボット制御サブシステムは，形状測定サブシステムからの情報を基に，不定形剛体の把持，運搬，据置を行う．

また，不定形剛体運搬ロボットシステムのアーキテクチャ設計時において，つぎの成果物が作成されている．

- 不定形剛体運搬ロボットシステムの仕様書
- ロボット制御サブシステム仕様書
- 形状測定サブシステム仕様書
- サブシステム間のインタフェース仕様書
- トレーサビリティマトリクス

表 3-2 に，不定形剛体運搬ロボットシステムのソースコードライン数およびモジュール数を示す．ロボット制御サブシステムおよび形状測定サブシステムの総ソースコードライン数が 20 万ライン弱であることから，不定形剛体運搬ロボットシステムは，規模の大きい組込みシステムであるといえる．また，ロボット制御サブシステムおよび形状測定サブシステムのモジュール（関数）数が 500 モジュール弱であることから，不定形剛体運搬ロボットシステムは，高機能な組込みシステムであるといえる．

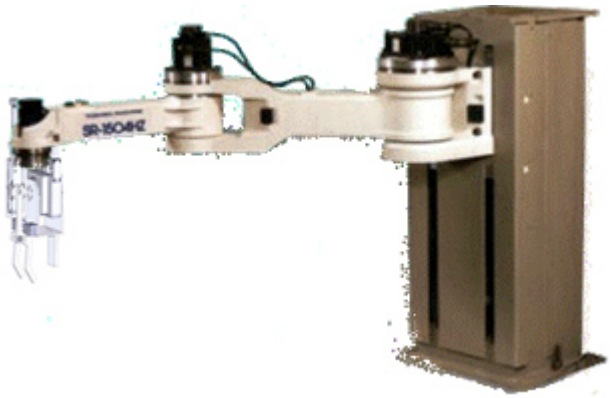


図 3-7　不定形剛体運搬ロボットシステム

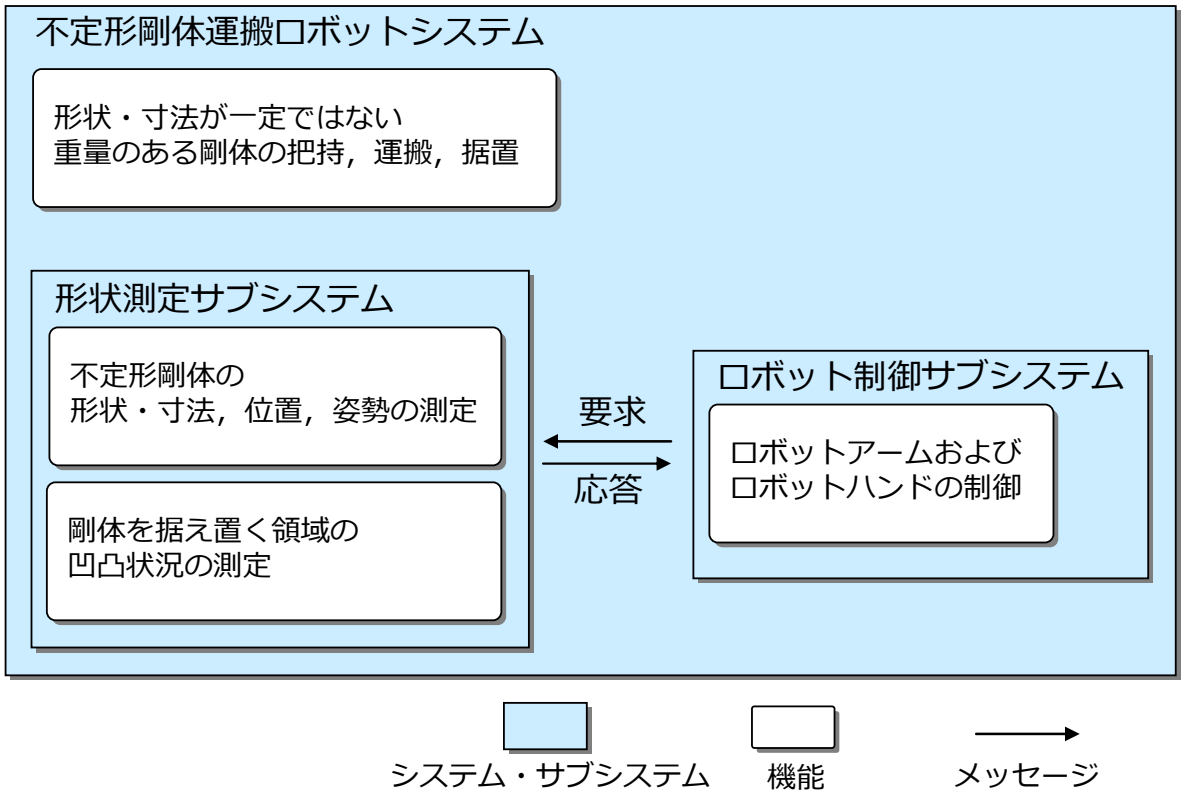


図 3-8　不定形剛体運搬ロボットシステムのシステム構成

表 3-2　不定形剛体運搬ロボットシステムのライン数およびモジュール数

	ロボット制御サブシステム	形状測定サブシステム
ソースコードライン数（C言語）	10,000	7,000
モジュール数	350	120

3.4.2　実験環境

評価実験において，モデル検査ツールが動作する計算機環境は，つぎのとおりである．

- クロック周波数：1.86GHz
- メモリ：2.96GB

また，評価実験では，モデル検査ツールとして UPPAAL を用いた．

3.4.3　提案手法の適用

(a) 構成要素の連携部分の抽出

トレーサビリティマトリクスを基に，ロボット制御サブシステムおよび形状測定サブシステムの仕様，サブシステム間のインタフェース仕様から，連携部分に関する仕様を抽出した．連携部分に関する仕様の抽出について，その一例を以下に示す．

つぎに示す仕様は，それぞれの仕様書からの抜粋で，設計情報を抽象化している．トレーサビリティマトリクスにおいて，不定形剛体運搬ロボットシステムの仕様 6.1.8.2.a は，ロボット制御サブシステムの仕様 3.1.1.1.a から 3.1.1.1.f，形状測定サブシステムの仕様 6.1.1.1.1.a から 6.1.3.1.a，ロボット制御サブシステム／形状測定サブシステム間インタフェース仕様の 4.1 から 4.2.4.1 に対応付けられる．

不定形剛体運搬ロボットシステム仕様

6.1.8.2.a　　一定の位置に据え置かれている剛体を見出し把持すること．

ロボット制御サブシステム仕様

3.1.1.1.a　コンソールからの剛体把持要求を待つこと.

3.1.1.1.b　剛体把持要求を受信した場合，剛体測定要求を作成すること.

3.1.1.1.c　作成した剛体測定要求を測定制御サブシステムに対して送信すること.

3.1.1.1.d　剛体測定要求を送信後,測定制御サブシステムからの剛体測定応答の受信待ちを行うこと.

3.1.1.1.e　形状測定サブシステムから剛体測定応答を受信した場合,剛体測定応答の内容を基にロボットアームおよびロボットハンドを制御し剛体の把持を行うこと.

3.1.1.1.f　剛体の把持を完了した場合，剛体把持要求の結果をコンソールに出力すること.

形状測定サブシステム仕様

6.1.1.1.1.a　ロボット制御サブシステムから剛体測定要求を受信すること.

6.1.1.1.2.a　剛体測定要求を受信した場合，剛体の測定を開始すること.

6.1.2.1.a　レーザスキャナを測定開始位置に移動すること.

6.1.2.1.b　レーザスキャナを用いて剛体を測定し剛体の標高データを取得すること.

6.1.2.1.c　剛体の標高データの取得時点におけるレーザスキャナの上下位置を取得すること.

6.1.2.1.d　剛体に対する 1 走査分の標高データの取得が終了した場合,レーザスキャナの上下位置を θ 度進めること.

6.1.2.1.e　レーザスキャナの上下位置が ψ から φ 度における標高データを測定すること.

6.1.2.1.f　レーザスキャナの上下位置,その位置における剛体の標高データがマトリクスになった測定 CSV データを作成すること.

6.1.3.1.a　測定 CSV データを基に剛体の測定データを算出すること.

6.1.1.2.1.a　剛体測定要求に対する測定処理が正常に終了した場合,ロボット制御サブシステムに対して測定データおよびエラー種別（正常）を送信すること.

6.1.1.2.1.b　剛体測定要求に対する測定処理においてエラーが発生した場合，ロボット制御サブシステムに対して該当するエラー種別を送信すること．

ロボット制御サブシステム／形状測定サブシステム間インタフェース仕様

4.1　物理インタフェース

4.2.2.1　剛体測定要求／応答シーケンス

4.2.3　データ定義

4.2.4.1　剛体測定要求／応答メッセージフォーマット

構成要素の連携部分に関する仕様の抽出について，ロボット制御サブシステム仕様における仕様 78 項目から 12 項目を抽出した，形状測定サブシステム仕様における仕様 58 項目から 23 項目を抽出した．また，サブシステム間のインタフェース仕様における仕様 26 項目から 22 項目を抽出した．

(b) 連携部分のプロパティの識別

トレーサビリティマトリクスを基に，不定形剛体運搬ロボットシステムのシステム仕様から，ロボット制御サブシステムおよび形状測定サブシステムによる連携のプロパティを識別した．連携部分のプロパティの識別について，その一例を示す．

3.4.3 (a)に示すロボット制御サブシステムおよび形状測定サブシステムの連携に関する仕様は，3.4.3 (a)で示す不定形剛体運搬ロボットシステムの仕様以外に，つぎの仕様に対しても対応付けられる．つぎに示す仕様は，不定形剛体運搬ロボットシステムの仕様書からの抜粋で，設計情報を抽象化している．

5.5.2　不定形剛体運搬ロボットシステムを構成する装置間におけるデータ送受信は要求／応答方式を採用すること．

6.3.2.c　コンソールからの動作停止要求に対して t ms 以内に不定形剛体運搬ロボットシステムの動作を停止すること．

3.4.3 (a)で示す，ロボット制御サブシステムおよび形状測定サブシステムの連携部分は，これらのシステム仕様を満たす必要がある．そこで，これらのシステム仕様を基に，ロボット制御サブシステムおよび形状測定サブシステムによる

連携部分のプロパティを抽出した．連携部分のプロパティをつぎに示す．P.1-5 および P.3-3 は，それぞれの性質に付与した番号である．

P.1-5　ロボット制御サブシステムは，形状測定サブシステムに対する剛体測定要求に対し，形状測定サブシステムから剛体測定結果もしくは剛体測定失敗の応答を受信すること．

P.3-3　形状測定サブシステムは，ロボット制御サブシステムによる測定停止要求の送信後 100ms 以内に測定処理を停止すること．

以上に示す手順により，連携部分のプロパティとして 23 項目を抽出した．

(c) モデル検査の実施

ロボット制御サブシステムおよび測定制御サブシステムの仕様，サブシステム間のインタフェース仕様から抽出した連携部分の仕様を基に，UPPAAL の表現形式に従い，連携部分のモデルを作成した．図 3-9 から図 3-11 に連携部分のモデルを示す．

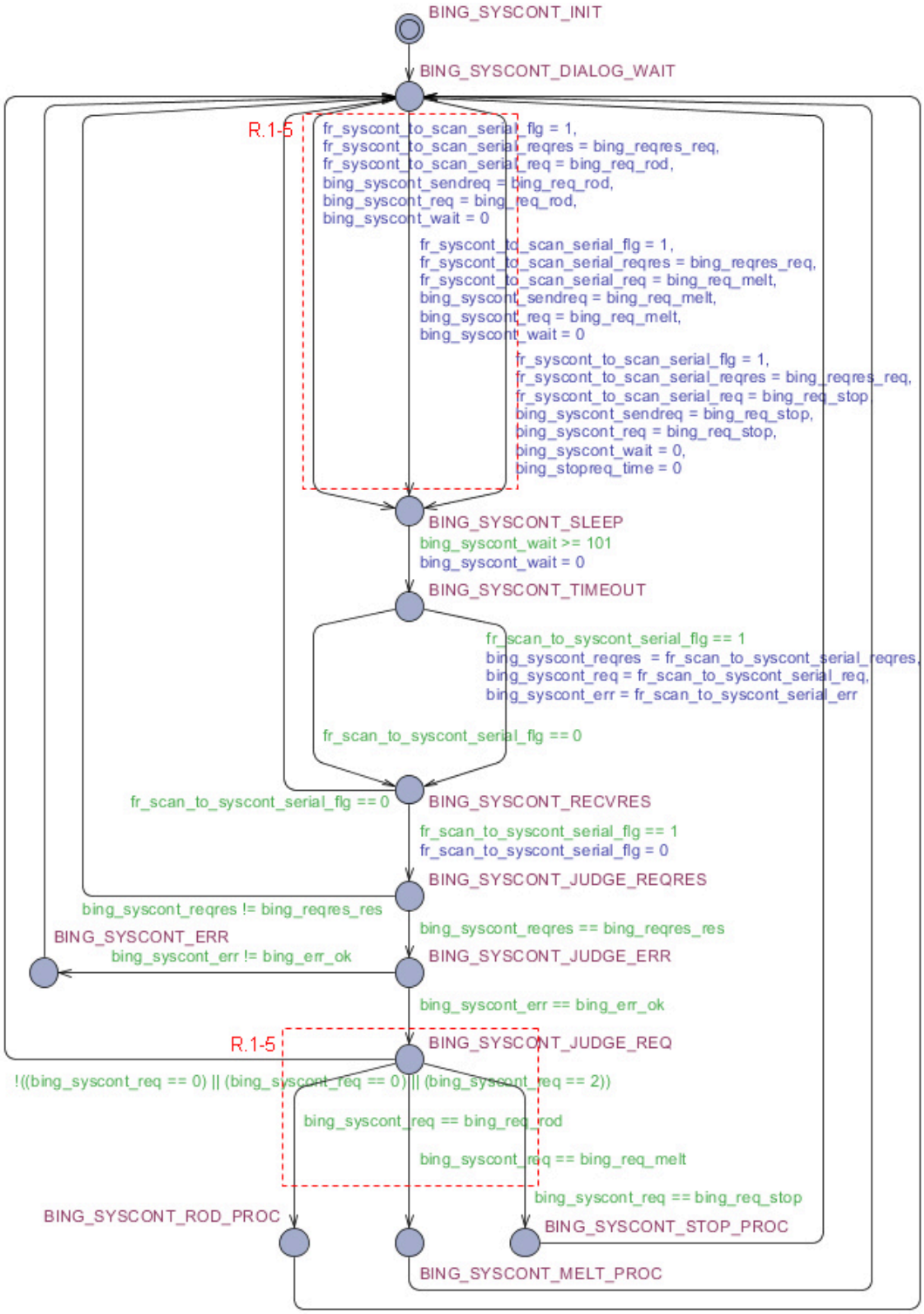


図 3-9　ロボット制御サブシステムにおける連携モデル

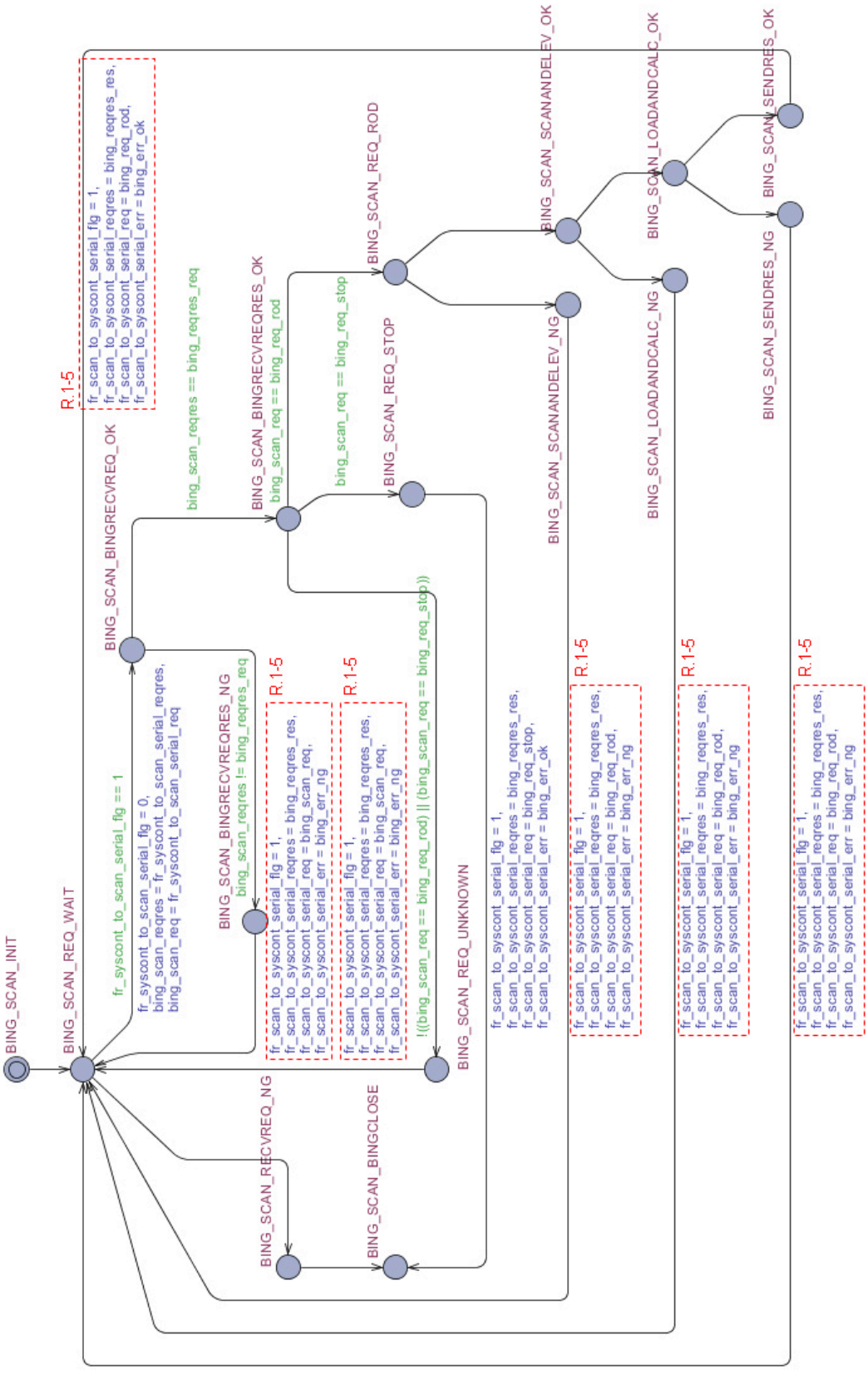


図 3-10　形状測定サブシステムにおける連携モデル

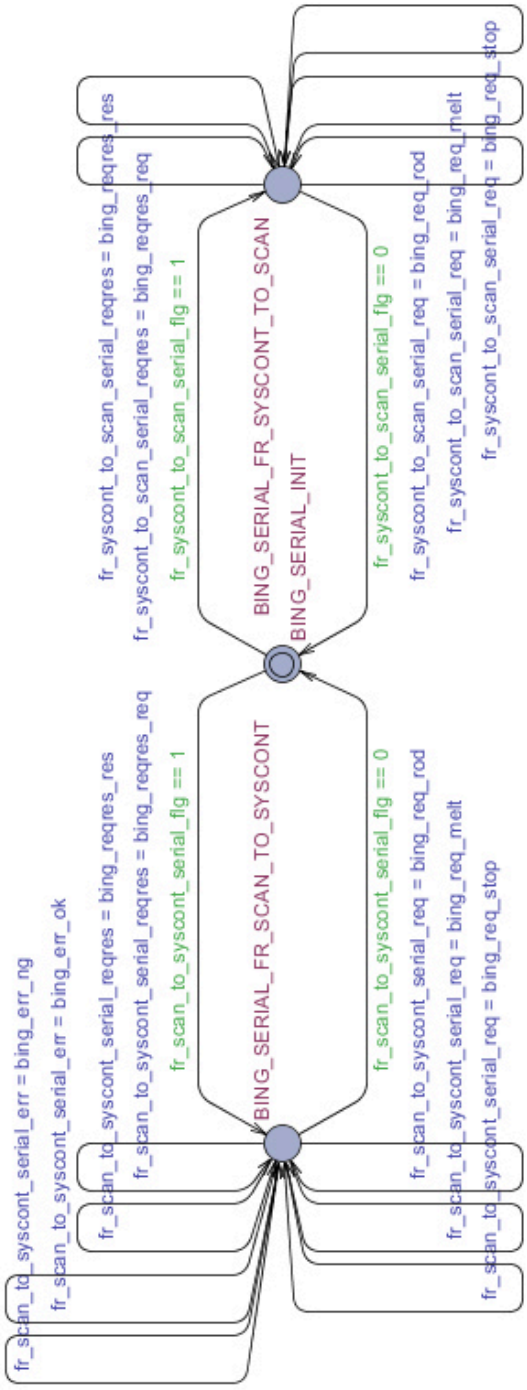


図 3-11　サブシステム間インタフェース仕様における連携モデル

作成したモデルは，連携部分において，ロボット制御サブシステムに対応するモデル（図 3-9），形状測定サブシステムに対応するモデル（図 3-10），サブシステム間のインタフェースに対応するモデル（図 3-11）である．UPPAAL による連携部分のモデルは，連携の状態遷移を表現する．状態遷移モデルを作成するにあたり，まず，連携に関する仕様の中から「～待ち」や「～中」などを識別する．3.4.3 (a)に示す形状測定サブシステムの連携に関する仕様 6.1.1.2.1.a「剛体測定要求に対する測定処理が正常に終了した場合，ロボット制御サブシステムに対して測定データおよびエラー種別（正常）を送信すること」の場合，その仕様には「剛体測定要求に対する測定処理を実施中」という振る舞いが含まれる．これが状態遷移モデルにおける状態となる．つぎに連携に関する仕様の中から，状態に関連するイベントを識別する．前述の連携に関する仕様 6.1.1.2.1.a の場合，「測定処理の正常終了」がイベントである．これが状態遷移モデルにおける状態の遷移条件となる．そして，連携に関する仕様の中から，状態遷移に関するアクションを識別する．前述の連携に関する仕様 6.1.1.2.1.a の場合，「ロボット制御サブシステムに対して測定データおよびエラー種別（正常）を送信」がアクションである．これが状態遷移モデルにおける状態遷移時に行う手続きとなる．

また，UPPAAL の表現形式に従い，ロボット制御サブシステムおよび形状測定サブシステムの連携部分のプロパティとして，23 ケースの検査式を作成した．前述の連携部分のプロパティである P.1-5 および P.3-3 に対応する検査式を，つぎに示す．F.1-5 および F.3-3 は，それぞれの検査式に付与した番号である．

F.1-5 *A[] (bing_syscont_sendreq == bing_req_rod) imply (bing_syscont_req == bing_req_rod)*

F.3-3 *A[] P_BING_SCAN.BING_SCAN_REQ_STOP imply (bing_stopreq_time <= 10)*

図 3-9 から図 3-11 に示す連携モデルおよび F.1-5 および F.3-3 を含む 23 ケースの検査式を基に，UPPAAL によるモデル検査を実施した．モデル検査の実施後，UPPAAL により出力される反例を分析した．分析の結果，F.1-5 および F.3-3 の検査式を含む 6 ケースの仕様の誤りを検出した．F.1-5 および F.3-3 に対応する連携部分の仕様の誤りをつぎに示す．また，図 3-12 に，その概要を示す．R.1-5

および R.3-3 は，F.1-5 および F.3-3 に対応する仕様の誤りに付与した番号である．R.1-5 については，図 3-9 および図 3-10 において，連携に関する誤りが混入した箇所を点線で示す．

R.1-5　メッセージおよび処理のタイミングによっては，n 回目に発行された剛体測定要求に対して n 回目より前に発行された剛体測定要求が返却される場合がある．

R.3-3　測定停止要求に対して 100ms 以内に測定処理が停止しない場合がある．また，測定停止要求に対して測定サブシステム自体が停止する場合がある．

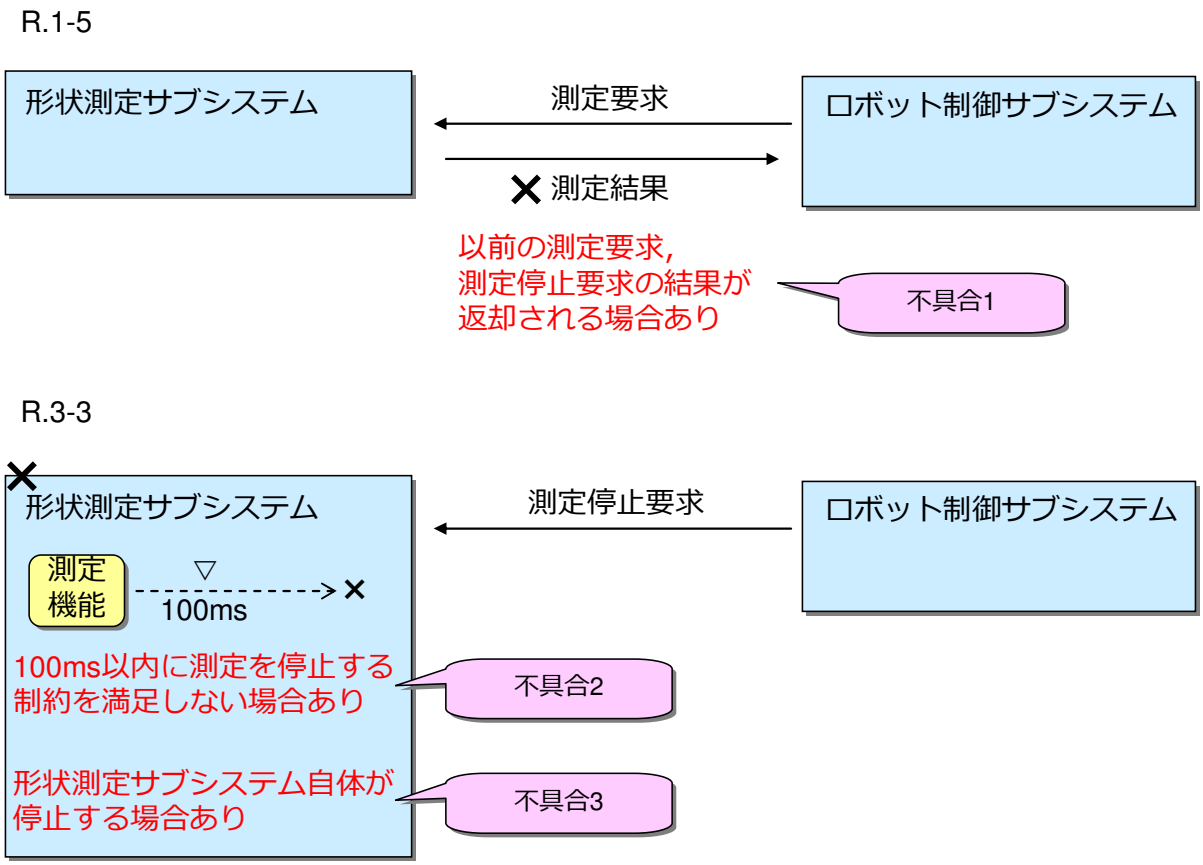


図 3-12　検出された連携部分の不整合

3.4.4　実験結果

提案手法における構成要素の連携部分の検証に着目し, 産業界の実例に対して提案手法を適用した．その結果，状態爆発を発生させることなく，構成要素の連携の正しさを検証することができた．また，検証においては，構成要素を組み合わせなければ発見できない不具合を検出することができた.

3.5　考察

3.4 で示した評価実験の結果を基に，提案手法の有効性を述べる．また，提案手法の課題を述べる.

3.5.1　提案手法の有効性

評価実験において, 不定形剛体運搬ロボットシステムという機能的に複雑な産業用ロボットに対して提案手法を適用した．その結果，状態爆発を発生させることなく，ロボット制御サブシステムおよび形状測定サブシステムの連携部分の不整合を検出することができた．特に，3.4.3 の R.1-5 および R.3-3 に示す連携部分の不整合については，プロパティが成り立つか否かをしらみつぶしに確認するモデル検査ならではの，人手では検出することが困難な不具合といえる.

今回の実験では評価しなかった, 提案手法における構成要素の検証については, 今後，十分な評価や考察を行う必要がある．しかし，構成要素の検証において，仮に状態爆発が発生した場合，その構成要素をさらに分割し，分割したそれらに対して提案手法を適用する方法が考えられる.

また，提案手法は，構成要素および構成要素の連携部分の単位で，状態遷移モデルを作成する．その際，対象を絞ってモデル化することで，モデルに誤りが混入することを低減できる可能性がある．システム全体をモデル化することは，それ自体が困難な作業である．一般に，モデル化する規模が大きくなると，モデル化時に考慮すべき条件が増える．その場合，モデルに誤りが混入する可能性が高くなる．モデル検査の対象を分割し，構成要素および構成要素の連携部分に対象を絞ってモデル検査するメリットは，このあたりにもある．これらの観点からも，複雑なシステムに対する提案手法の適用性は高いと考える.

3.5.2　提案手法の課題

3.5.1 において，提案手法の有効性を確認することができた．本項では，提案手法に関する今後の課題を述べる．

提案手法を形式化し，手法の健全性を厳密に示す必要がある．提案手法では，構成要素および構成要素の連携部分をモデル検査することで，システム全体を検証する．提案手法のこのアプローチについて，暗にはその正しさを理解することができる．しかし，明にはその正しさを示していない．また，提案手法を形式化し，実施者への依存度が低い手法を構築する必要がある．提案手法が形式化されていないことで，提案手法の適用において，属人性が入り込む可能性がある．手法の形式化は，手法に対する計算機支援を考えた場合に，必要不可欠である．

また，提案手法において，トレーサビリティマトリクスの信頼性を確保する必要がある．提案手法では，構成要素の連携部分に着目したモデル検査を実施する．その際，システムのアーキテクチャ設計時に作成されるトレーサビリティマトリクスを用いて，構成要素の仕様から連携部分を抽出する．そのため，トレーサビリティマトリクスに誤りがある場合，構成要素の連携部分を正しく抽出できない．本課題の対策としては，システムおよび構成要素の仕様を，形式手法の一つである形式仕様記述で形式化し，それらに対応するトレーサビリティマトリクスを自動で生成する方法が考えられる．また，形式化されたシステムと構成要素の仕様および形式化されたトレーサビリティマトリクス間の整合性を検証し，トレーサビリティマトリクスの信頼性を確保する方法が考えられる．トレーサビリティマトリクスの信頼性を確保することで，提案手法の手法としての信頼性を向上させることができる．また，形式化された仕様書および検証されたトレーサビリティマトリクスを用いて，連携部分の自動抽出および状態遷移モデルの自動生成を行うことができる可能性がある．それにより，連携部分における状態遷移モデルの正しさの確保が期待できる．

3.6 まとめ

本章では，システムにおける構成要素および構成要素の連携部分を，それぞれ個別にモデル検査する手法を提案した．提案手法では，トレーサビリティマトリクスを用いて，構成要素の仕様書および構成要素間のインタフェースの仕様書から，連携部分を漏れなく抽出する．提案手法の有効性を確認するために，不定形剛体運搬ロボットシステムという産業界の実例に対して，提案手法を適用した．その結果，不定形剛体運搬ロボットシステムにおける構成要素の連携部分について，状態爆発を発生させることなく，構成要素を組み合わせなければ発見できない不具合を検出することができた．

第4章　MWF 非起動および MNWF 起動の観点を用いた安全性プロパティの導出

本章では，「MWF 非起動」および「MNWF 起動」の観点を用いて，モデル検査に適用可能な，具体的な安全性プロパティを導出する手法を述べる．本章で述べる手法は，第 1 章で述べた課題 2 を解決する．

4.1　はじめに

複雑なシステムの信頼性を向上させる技術として，モデル検査が注目されている．モデル検査とは，検査対象の状態遷移が，検査対象として満たすべき性質（プロパティ）を満たすか否かを，網羅的に検証する技術である．プロパティは，検査対象の上位仕様，過去の事故・不具合などから導出される．プロパティには，活性プロパティおよび安全性プロパティがある．本論文では，モデル検査における安全性プロパティの導出に着目する．安全性プロパティとは，「システムにとって望ましくない事象が決して起こらない」性質である．安全性プロパティの基になるシステムの望ましくない事象は，非形式的かつ抽象的に表現される．無線踏切システム[65][66]の場合，システムの望ましくない事象は，「列車がロードユーザ（自動車や歩行者など）と衝突する」などである．モデル検査では，システムの望ましくない事象を基に，モデル検査に適用可能な具体的な安全性プロパティが導出される必要がある．

複雑なシステムのモデル検査において，具体的な安全性プロパティが導出される際には，つぎに示す 2 つの課題がある．

- 導出すべき具体的な安全性プロパティの識別に抜けが生じる可能性がある．
- 安全性プロパティの具体化が不十分な場合がある．

図 4-1 に，具体的な安全性プロパティを導出する際の課題を示す．一般に，モデル検査の対象が単純なシステムの場合，望ましくない事象の具体化にあたり，考慮すべきシステムの機能や条件は単純である．望ましくない事象の具体化が適切に行われ，モデル検査に適用可能な具体的な安全性プロパティが導出され

る場合が多い．本論文における望ましくない事象の具体化とは，抽象的に表現された望ましくない事象を，システムの仕様などを用いて具体的に言い表すことを意味する．しかし，システムが複雑になるにつれ，考慮すべきシステムの機能や条件が複雑になる．その結果，望ましくない事象の具体化において考慮が不足し，具体的な安全性プロパティに抜けが生じる可能性がある．また，モデル検査に用いる安全性プロパティとして，具体化が不十分な場合がある．本論文において安全性プロパティの具体化が不十分とは，モデル検査のモデルの記述と比較して，安全性プロパティの記述が抽象的であることを意味する．

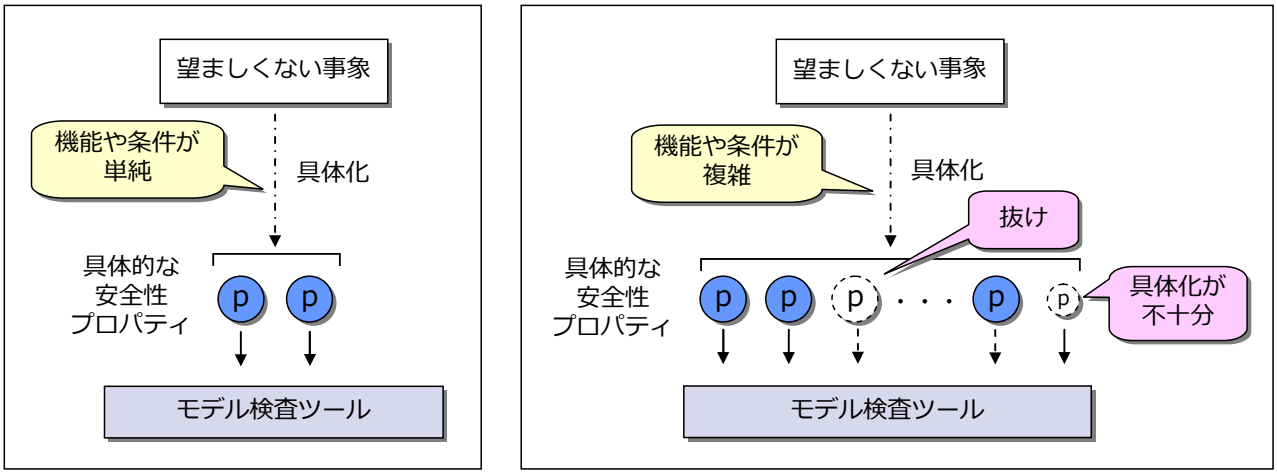


図 4-1　具体的な安全性プロパティ導出の課題

本論文では，「MWF 非起動」および「MNWF 起動」の観点で，システムの望ましくない事象を具体化し，モデル検査に適用可能な具体的な安全性プロパティを導出する手法を提案する．MWF とは，その意図しない非起動が，システムの望ましくない事象を引き起こす可能性がある機能である[10]．MNWF とは，その意図しない起動が，システムの望ましくない事象を引き起こす可能性がある機能である[10]．提案手法では，導出されるプロパティの網羅性向上に寄与すると考えられる「MWF 非起動」および「MNWF 起動」の 2 つの観点を用いて，システムの望ましくない事象が具体化される．また，提案手法では，これらの観点に基づき，望ましくない事象の具体化が繰り返し行われ，より具体的な事象が導出される．望ましくない事象の具体化の後，具体化された事象の否定 (¬) を取ることで，具体的な安全性プロパティが導出される．本論文では，「MWF 非起動」および「MNWF 起動」の観点によるプロパティの具体化は，項書換え系[67]の考えに基づく知識の書換えによって形式化されている．提案手法において，項書換え系の考えが採用される理由は，提案手法における事象の具体化を，厳密かつ明確に定義するためである．また，項書換え系の考えに基づく知識の書換えルールを，図 4-9 および図 4-10 で示すように図示した場合，提案手法の実施者にとって，具体化の手順が直感的で理解しやすいためである．提案手法では，モデル検査の対象となるシステムの仕様から，システムの望ましくない事象に対する MWF および MNWF が識別される．識別されたそれらは，知識として Data Base（以降，DB という）に登録される．DB に登録されている MWF および MNWF に関する知識を用いて，知識の書換えが行われる．なお，本論文においては，提案手法の適用に必要なシステムの機能およびその起動条件は，システムの仕様書に抜けなく記述されているものとする．また，モデル検査の状態遷移モデルで用いられる変数は，提案手法による具体的な安全性プロパティで用いられるシステムの変数と対応付けられているものとする．

提案手法の有効性を確認するために，無線踏切システムを適用対象とする，提案手法の評価実験を行った．実験の結果，モデル検査の具体的な安全性プロパティの導出において，プロパティの網羅性および具体性が向上することを確認した．

本章の構成はつぎのとおりである．4.2 において，関連研究を述べる．4.3 において，提案手法を詳述する．4.4 において，提案手法に対する評価実験の結果を示す．4.5 において，提案手法の有効性を考察する．最後に 4.6 において，本論

文をまとめる．

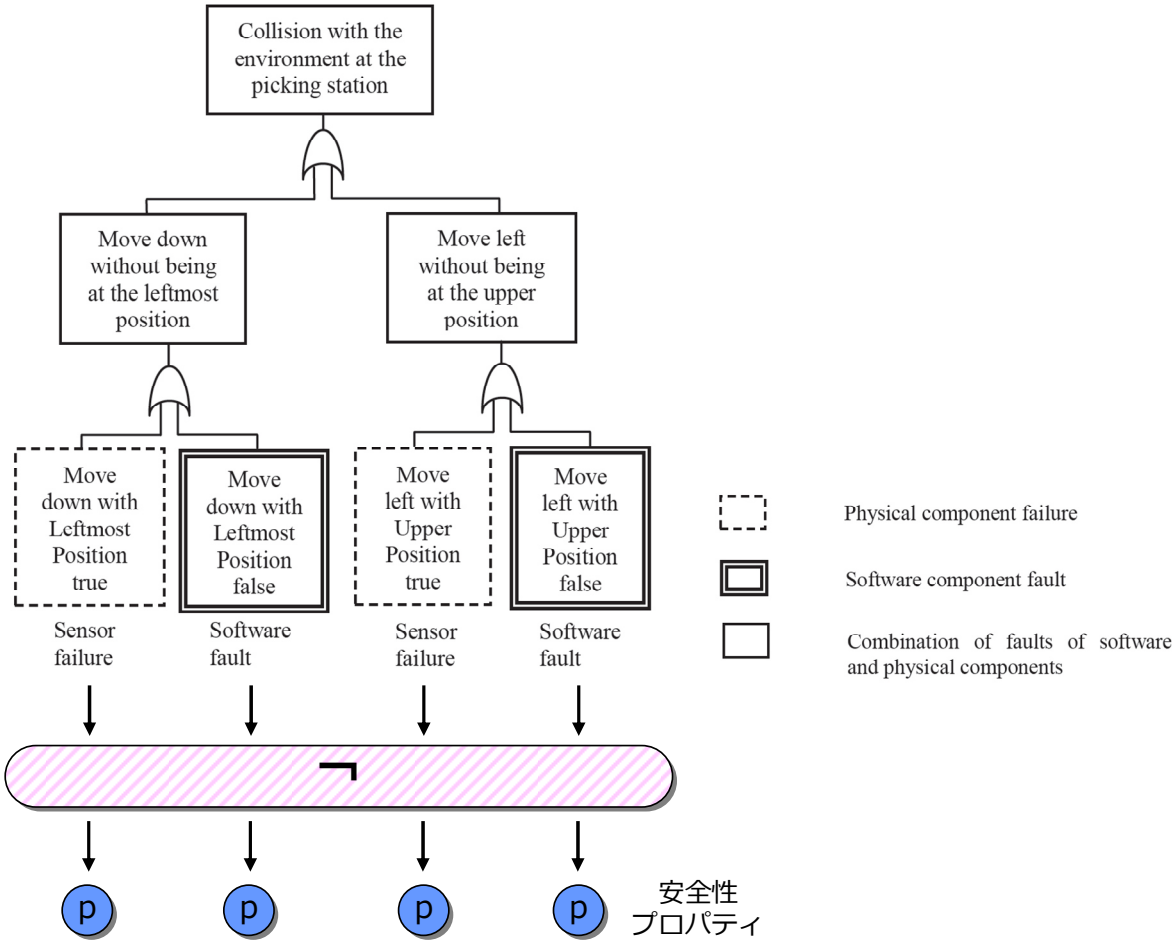
4.2 関連研究

モデル検査における安全性プロパティを導出する方法として, FTA およびゴール指向要求分析を用いる方法が提案されている．

Henry ら，Barragan らは，それぞれ[68]，[69]において，Fault Tree Analysis（以降，FTA という）[70][71][72]を用いて，安全性プロパティを導出する方法を提案している. 図 4-2 に, FTA を用いて安全性プロパティを導出する手法を示す．FTA とは，システムの望ましくない事象を頂上事象とする Tree を作成し，頂上事象の原因となる事象を識別する解析手法である．[68]および[69]では，システムの望ましくない事象を頂上事象として FTA を行い，その結果の否定を取ることで安全性プロパティを導出している．FTA を用いて，望ましくない事象の原因となる事象を演繹的に導出していく．それにより，安全性プロパティとして妥当な事象を導出することができる．しかし，[68]および[69]では，Fault Tree の作成において，親事象から子事象を導出するルールが定義されていない．そのため，安全性プロパティの導出が実施者の経験やスキルに依存してしまい，導出する安全性プロパティに抜けが生じる可能性がある．また，安全性プロパティの具体化が不十分になる可能性がある．手法については，非形式的に定義されている．

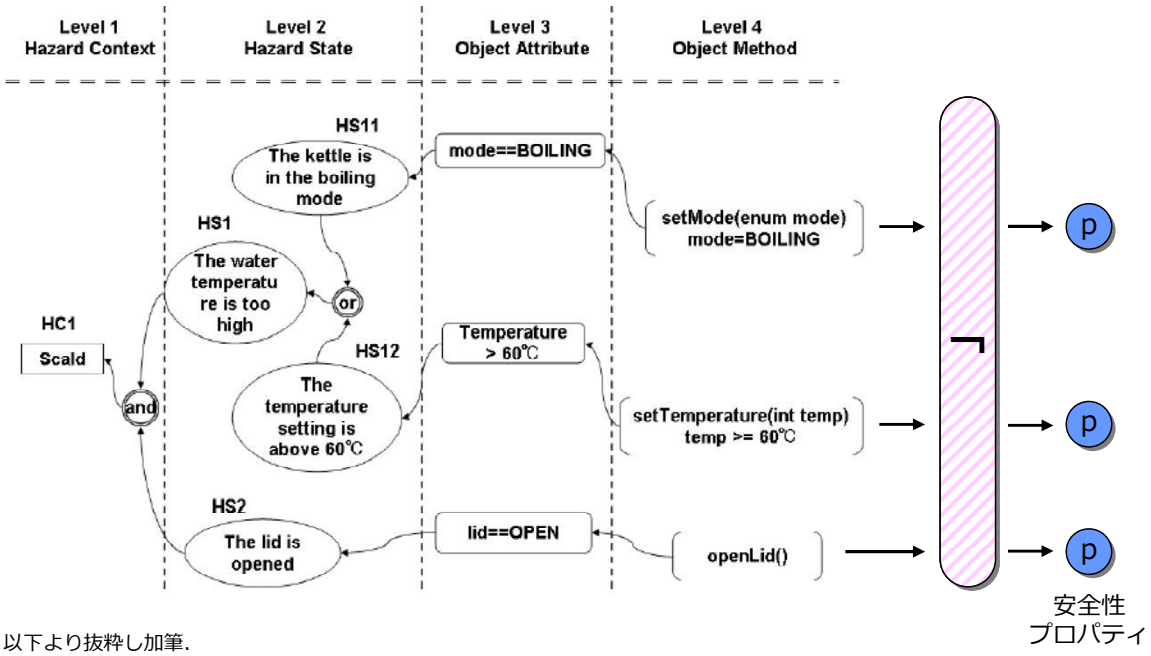
Yan らは，[73]において，ゴール指向要求分析[74][75]の考えに基づく Hazard Analysis Model（以降，HAM という）を用いて，安全性プロパティを導出する方法を提案している．図 4-3 に，ゴール指向要求分析を用いて安全性プロパティを導出する手法を示す．ゴール指向要求分析とは，システムが達成すべきゴールを分解し，より具体的なゴールを識別する解析手法である．[73]では，Level 1 の Hazard Context，Level 2 の Hazard State，Level 3 の Object Attribute，Level 4 の Object Method という 4 レベルの階層を持つ HAM を考案している．HAM の中で最も抽象度の高い Level 1 の Hazard Context に対して，システムの望ましくない事象を与えている. その事象を Level 2, Level 3, Level 4 を通して具体化し，その結果の否定を取ることで，安全性プロパティを導出している．HAM を用いて Level 毎の段階的な具体化を行う．それにより，のぞましくない事象を確実に具体化することができる．また，HAM の Level 2 において，論理記号の AND お

よび OR を用いて, 望ましくない事象の原因となる事象を演繹的に導出していく．それにより，安全性プロパティとして妥当な事象を導出することができる．しかし，[73]では，Level 2 の Hazard State において，Level 1 の Hazard Context の望ましくない事象を基に，具体的な事象を導出するルールが定義されていない．そのため，安全性プロパティの導出が実施者の経験やスキルに依存してしまい，導出する安全性プロパティに抜けが生じる可能性がある．手法については，非形式的に定義されている．



以下より抜粋し加筆.
Henry, S. and Faure, M. J.: Elaboration of invariants safety properties from fault-tree analysis, In Proceeding of IMACS-IEEE Computational Engineering in Systems Applications (2003).

図 4-2　FTA を用いた安全性プロパティの導出



以下より抜粋し加筆.
Yan, B., Nakamura, M. and Matsumoto, K.:
Deriving Safety Properties for Home Network System Based on Goal-Oriented Hazard Analysis Model,
International Journal of Smart Home, Vol. 3, No. 1, pp.67–79 (2009).

図 4-3　ゴール指向要求分析を用いた安全性プロパティの導出

一方，提案手法では，「MWF 非起動」および「MNWF 起動」の 2 つの観点を用いた系統的な方法により，システムの望ましくない事象を具体化する．それにより，導出する望ましくない事象の網羅性の向上を図る．また，システムの仕様書から MWF および MNWF が識別可能な限り，「MWF 非起動」および「MNWF 起動」の観点による具体化を繰り返し行う．提案手法における「MWF 非起動」および「MNWF 起動」の観点による具体化は，形式的に定義されている．

表 4-1 に，安全性プロパティの導出に関する先行研究と提案手法の比較結果をまとめる．

表 4-1　安全性プロパティの導出に関する先行研究と提案手法の比較

	FTA法 [※1][※2]	ゴール指向要求分析法 [※3]	提案手法
安全性プロパティの網羅性	× Fault Treeで 親事象から子事象を 導出するルールが未定義	× HAMのLevel 2における 導出ルールが未定義	○ MWF非起動/MNWF起動の 観点による系統的な具体化
安全性プロパティの具体性	× Fault Treeで 親事象から子事象を 導出するルールが未定義	○ HAMのLevel 1～4での 段階的な具体化.	○ MWF非起動/MNWF起動の 観点による具体化の繰り返し
安全性プロパティの妥当性	○ 望ましくない事象を基に その原因を導出していく 演繹的なアプローチ	○ HAMのLevel 2における 望ましくない事象を基に その原因を導出していく 演繹的なアプローチ	○ MWF非起動/MNWF起動の 観点による具体化アプローチ
手法の形式性	× 手法を非形式的に定義	× 手法を非形式的に定義	△ MWF非起動/MNWF起動の 観点による具体化を 形式的に定義
属人性	× Fault Treeの作成が 実施者に依存	× HAMの作成が 実施者に依存	△ 仕様書からの MNWFの識別が 実施者に依存

※1 Henry, S. and Faure, M. J.: Elaboration of invariants safety properties from fault-tree analysis, In Proceeding of IMACS-IEEE Computational Engineering in Systems Applications (2003).

※2 Barragan, S. I. and Faure, M. J.: From Fault Tree Analysis to Model Checking of logic Controllers, In Proceeding of 16th International Federation of Automatic Control World Congress (2005).

※3 Yan, B., Nakamura, M. and Matsumoto, K.: Deriving Safety Properties for Home Network System Based on Goal-Oriented Hazard Analysis Model, International Journal of Smart Home, Vol. 3, No. 1, pp.67–79 (2009).

4.3 提案手法

モデル検査における具体的な安全性プロパティは, システムの望ましくない事象の具体化によって導出される．図 4-4 に，提案手法を示す.

システムの非形式的な望ましくない事象が提案手法に与えられる. 望ましくない事象の具体化は,「MWF 非起動」および「MNWF 起動」の 2 つの観点に基づき行われる．具体化においては,「MWF 非起動」に関する複数の事象が導出される場合がある．事象が 1 つも導出されない場合もある.「MNWF 起動」に関する事象も同様である.

「MWF 非起動」および「MNWF 起動」の観点による具体化は，項書換え系の考えに基づく知識の書換えによって形式化されている．提案手法では，望ましくない事象は, 知識として捉えられる. 知識の書換えにより導出される事象も, 知識として捉えられる．それらの知識に対しても，書換えが行われる．それにより，事象の具体化が繰り返される．知識の書換えにおいて，書換えに必要な知識は，MWF DB および MNWF DB から取得される．提案手法では，システムの仕様書から識別された，望ましくない事象に対する MWF および MNWF なども，知識として捉えられる．MWF DB および MNWF DB には，それらの知識が格納される.

事象の具体化は, 知識に対して書換えのルールが適用されなくなった時点, 言い換えると,「MWF 非起動」に関する事象，または「MNWF 起動」に関する事象が導出されなくなった時点で終了する．その後，与えられた望ましくない事象から見て，末端に相当する事象（図 4-4 の点線で囲んだ事象）の否定（¬）がとられる．そして，安全性プロパティが CTL で表現される場合は A（all）および G（globally），LTL で表現される場合は G（globally）という時相論理演算子が, 否定の事象に対して付与され, 具体的な安全性プロパティが導出される.

4.3.1 において, 提案手法のプロセスを述べる. 4.3.2 において, MWF, MNWF,「MWF 非起動」,「MNWF 起動」の 4 つの概念を定義する．4.3.3 において，システムの仕様書から MWF および MNWF を識別する方法，識別された MWF および MNWF を，MWF DB および MNWF DB に登録する方法を述べる．4.3.4 において，書換えルールによる望ましくない事象の具体化を述べる．4.4.5 において，安全性プロパティの生成方法を述べる.

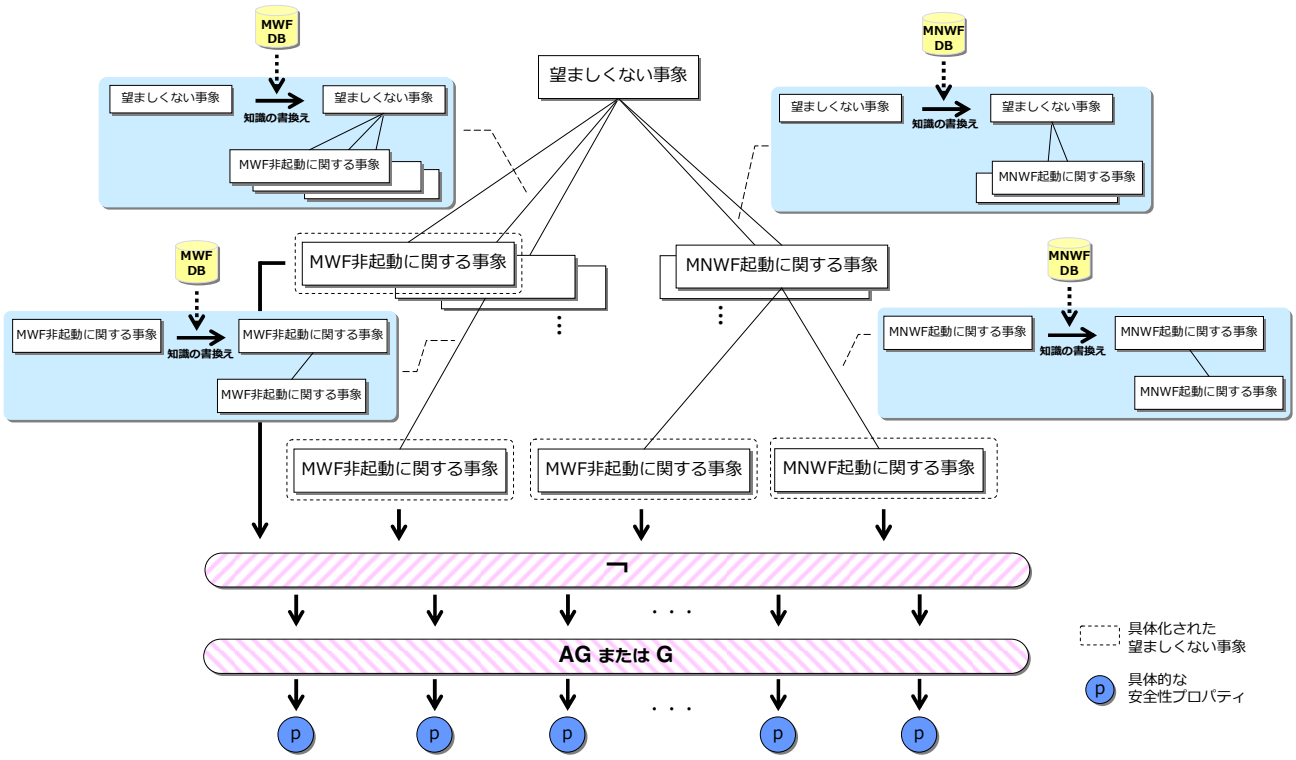

MWF DB
望ましくない事象
知識の書換え
望ましくない事象
MWF非起動に関する事象
望ましくない事象
MNWF DB
望ましくない事象
知識の書換え
望ましくない事象
MNWF起動に関する事象
MWF DB
MWF非起動に関する事象
MWF非起動に関する事象
知識の書換え
MWF非起動に関する事象
MWF非起動に関する事象
MNWF起動に関する事象
MNWF DB
MNWF起動に関する事象
知識の書換え
MNWF起動に関する事象
MNWF起動に関する事象
MWF非起動に関する事象
MWF非起動に関する事象
MNWF起動に関する事象
AG または G
p
具体化された
望ましくない事象
具体的な
安全性プロパティ

図 4-4　提案手法

4.3.1 提案手法のプロセス

提案手法に対して，システムの非形式的な望ましくない事象が与えられる．これを基に，提案手法では，モデル検査に入力可能な安全性プロパティが導出される．図 4-5 に，提案手法のプロセスを示す．以下の記述における番号（1.から4.）は，図 4-5 のブロック内に示される各番号に対応する．

1. 望ましくない事象に対する MWF および MNWF が，提案手法の適用対象となるシステムの仕様書から識別される. 識別された MWF および MNWF が，それぞれ MWF DB および MNWF DB に登録される．これらの識別と DB への登録方法については，4.3.3 に詳述する．
2. 項書換え系の考えに基づく知識の書換えルールにより，望ましくない事象が具体化される．書換えルールを適用可能な望ましくない事象が存在する場合，書換えルールの 1 つによって知識の書換えが行われる．その際，書換えルールは，MWF DB または MNWF DB に登録されている MWF や MNWF に関する知識を基に，望ましくない事象を「MWF 非起動」および「MNWF 起動」の観点に基づく事象に具体化する．これらの具体化は，3.で述べるように，繰り返し行われる．書換えルールによる望ましくない事象の具体化については，4.3.4 に詳述する．
3. 2.において，「MWF 非起動に関する事象」，または，「MNWF 起動に関する事象」が導出された場合，1.の直前に戻る．その際，「MWF 非起動に関する事象」は，「望ましくない事象」に置き換えられる．ここでの置き換えとは，導出された「MWF 非起動に関する事象」のラベルが「望ましくない事象」に置き換えられることを指している.「MWF 非起動に関する事象」および「望ましくない事象」の内容そのものは同じである．また，「MNWF 起動に関する事象」についても同様に，「望ましくない事象」に置き換えられる．それらにより，導出のプロセスが繰り返される．2.の結果によっては，複数の「MWF 非起動に関する事象」，または，「MNWF 起動に関する事象」，つまり，複数の「望ましくない事象」が導出される可能性がある．その場合，1.は，それらのすべての「望ましくない事象」を入力として実施される．2.において，「MWF 非起動に関する事象」または「MNWF 起動に関する事象」が 1 つも導出されなかった場合，事象の具体化が終了する．
4. 具体化された望ましくない事象の否定（¬）がとられる．否定化された事

象に対して，CTL の場合は AG，LTL の場合は G という時相論理演算子が付与される．それにより，具体的な安全性プロパティが生成される．具体的な安全性プロパティの生成については，4.3.5 に詳述する．

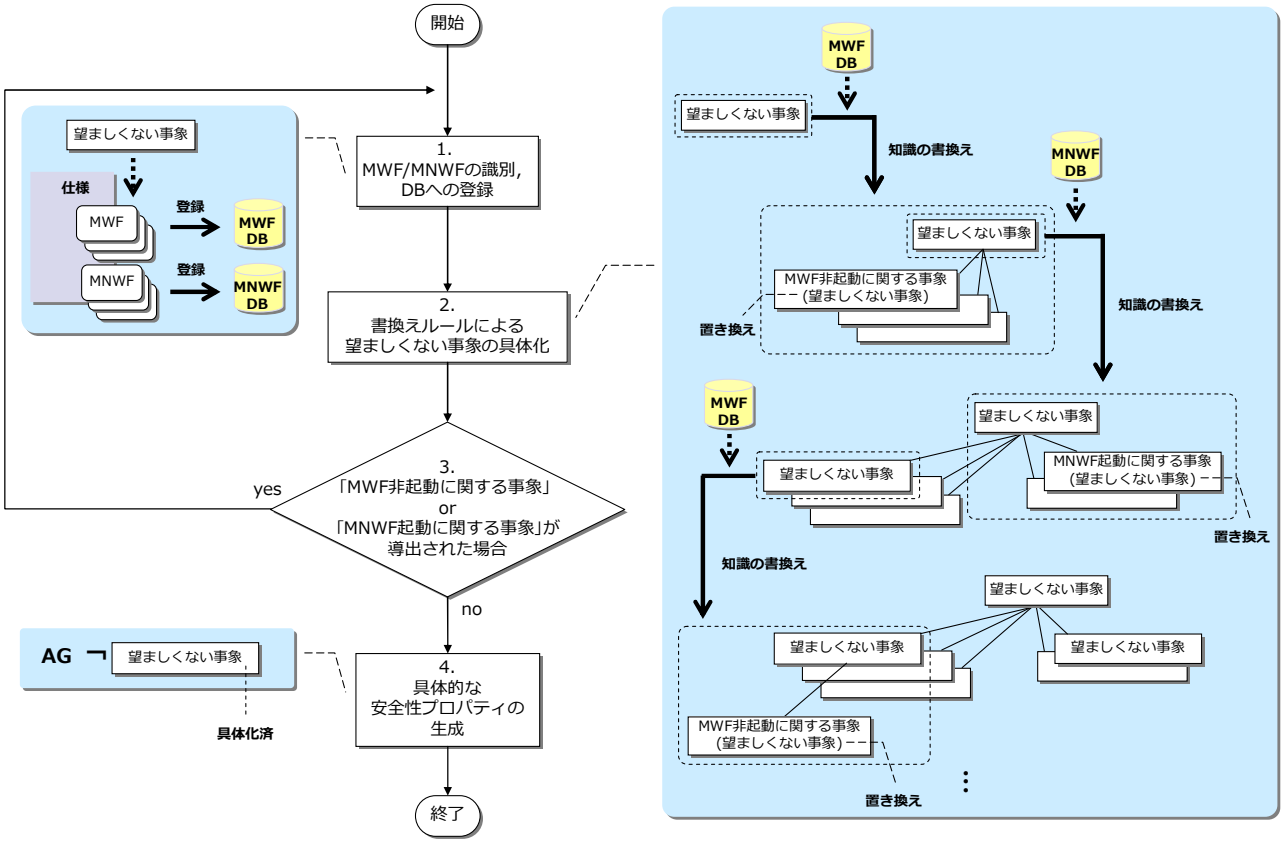


図 4-5　安全性プロパティ導出のプロセス

4.3.2 MWF，MNWF，MWF 非起動，MNWF 起動の定義

F を提案手法の適用対象となるシステムの機能の集合，S をシステムの状態の集合，U をこれらの直積集合 F×S とする．そして，以下のように，U 上の述語 1)から 3)を定義する．

1) Active(f,s) : 機能 f∈F は，システム状態 s∈S において active である．ここで，active とは，f が動作していることを表す.
2) MW_Ctx(f,s) : 機能 f が，システム状態 s において active でなければ，望ましくない事象が生じる可能性がある．
3) MNW_Ctx(f,s) : 機能 f が，システム状態 s において inactive でなければ，望ましくない事象が生じる可能性がある．ここで，inactive とは，f が動作していないことを表す.

これらの述語を用いて，まず，MWF および MNWF を，つぎのように定義する．「ある機能 f∈F が MWF である」とは，MW_Ctx(f,s)が true となる s∈S が，少なくとも 1 つ存在することであるとする．その場合，この定義は，変域 F から値域{true,false}への論理関数 MWF(f)によって，以下のように形式的に表現される．

MWF(f): f∈F → ∃s. MW_Ctx(f,s) の真理値 (1)

また，「ある機能 f∈F が MNWF である」とは，MNW_Ctx(f,s)が true となる s∈S が，少なくとも 1 つ存在することであるとする．その場合，この定義は，変域 F から値域{true,false}への論理関数 MNWF(f)によって，以下のように形式的に表現される．

MNWF(f): f∈F → ∃s. MNW_Ctx(f,s) の真理値 (2)

つぎに，「MWF 非起動」および「MNWF 起動」を定義する．「MWF 非起動」とは，s において f が active でなければ，望ましくない事象が生じる可能性があるにもかかわらず，f は active でない状況をいう．このとき，「MWF 非起動」は，以下のように形式的に表現される．

MWF 非起動(f,s) ≡ MW_Ctx(f,s)∧¬Active(f,s) (3)

無線踏切システムを例にいえば，「MWF 非起動」は，「(TrPos == appr_cr)∧¬(BaSts == closed)」となる．「TrPos == appr_cr」は，「The train approaches the crossing.」を意味する．「BaSts == closed」は，「The barrier is closed.」を意味する．また，また，「MNWF 起動」とは，s において f が inactive でなければ，望ましくない事象が生じる可能性があるにもかかわらず，f は active である状況をいう．この

とき，「MNWF 起動」は，以下のように形式的に表現される．

MNWF 起動(f,s) ≡ MNW_Ctx(f,s)∧Active(f,s) (4)

無線踏切システムを例にいえば,「MNWF 起動」は,「¬(TrPos == pass_cr)∧(BaSts == open)」となる．「TrPos == pass_cr」は，「The train passes the crossing.」を意味する．「BaSts == open」は，「The barrier opens.」を意味する．

4.3.3 MWF，MNWF の識別と DB への登録

図 4-6 に，望ましくない事象に対する MWF および MNWF の識別，および，識別された MWF および MNWF の DB への登録のプロセスを示す．提案手法の適用開始時点において，予め，提案手法に与えられるシステムの望ましくない事象は 1 つである．その後，望ましくない事象の具体化の過程において,「MWF 非起動」および「MNWF 起動」の観点により導出される事象も，望ましくない事象として取り扱われる（4.3.1 節参照）．それらの望ましくない事象が，図 4-6 のプロセスに入力される．以下の記述における番号（1.から 4.）は，図 4-6 のブロック内に示される各番号に対応する．図 4-6 の「システムの仕様書に定義されている機能のループ」の n は，システムの仕様書に定義されている機能の総数を表す．1.から 4.の処理は，システムの仕様書に定義されている，それぞれの機能 fi∈F に対して実施される．

1. fi が，望ましくない事象に対する MWF であるか否かが判定される．具体的には，システムの仕様書を基に，fi が active でなければ，望ましくない事象が生じる可能性があるシステム状態 s∈S が存在するか否かが確認される．そのような s が存在する場合，2.が実施される．fi に対する s は，複数存在することもある．s が存在しない場合，2.はスキップされる．
2. MWF が,「MWF 非起動」の際のシステム状態と共に，MWF DB に登録される．具体的には，1.で識別された fi および s が，それぞれ，MWF，および，fi を MWF にさせるシステム状態として，MWF DB に登録される．無線踏切システム[65][66]を例にいえば，図 4-6 の場合，望ましくない事象の「Collision」に対する MWF として,「(BaSts == closed)」が識別されている．この機能を MWF にさせるシステム状態として,「(TrPos == appr_cr)」が識別されている．これらの「(BaSts == closed)」および「(TrPos == appr_cr)」の組が，望ましくない事象と紐付けされた形で MWF DB に登録される．

fi に対する s は，複数存在することがある．その場合は，MWF DB に対して，fi とそれぞれの s の組が登録される．

3. fi が，望ましくない事象に対する MNWF であるか否かが判定される．具体的には，システムの仕様書を基に，fi が inactive でなければ，望ましくない事象が生じる可能性がある s が存在するか否かが確認される．そのような s が存在する場合，4.が実施される．fi に対する s は，複数存在することもある．s が存在しない場合，4.はスキップされる．
4. MNWF が，「MNWF 起動」の際のシステム状態と共に，MNWF DB に登録される．具体的には，3.で識別された fi および s が，それぞれ，MNWF，および，fi を MNWF にさせるシステム状態として，MNWF DB に登録される．図 4-6 の場合，望ましくない事象の「Collision」に関する MNWF として，「(BaSts == open)」が識別されている．この機能を MNWF にさせるシステム状態として，「¬(TrPos == pass_cr)」が識別されている．これらの「(BaSts == open)」および「¬(TrPos == pass_cr)」の組が，望ましくない事象と紐付けされた形で MNWF DB に登録される．fi に対する s は，複数存在することがある．その場合は，MNWF DB に対して，fi とそれぞれの s の組が登録される．

図 4-7 に，MWF DB の構造を示す．図 4-7 に示す例は，図 4-6 の無線踏切システムの例に対応している．MWF DB は，項「望ましくない事象」，および，項「MWF」，項「システム状態」，項「MWF 非起動に関する事象」で構成される．項「MWF 非起動に関する事象」の知識は，4.3.4 で述べる MWF DB の関数「getMWFdeact()」の実行時に生成される．その際，その知識は，4.3.2 の(3)に従い，項「MWF」，および，項「システム状態」の知識を基に生成される．生成された「MWF 非起動に関する事象」の知識は，MWF DB に登録される．MWF DB に対する MWF およびシステム状態の登録は，「MWF 非起動に関する事象」，または，「MNWF 起動に関する事象」が置き換えられた，望ましくない事象に対しても繰り返し行われる．そのため，MWF DB では，図 4-7 に示すように，複数の望ましくない事象の行が登録される．図 4-7 の矢印は，「MWF 非起動に関する事象」が望ましくない事象に置き換えられ，それに対する MWF およびシステム状態が，MWF DB に登録されていることを示している．
図 4-8 に，MNWF DB を示す．MNWF DB の考え方は，MWF DB と同様である．

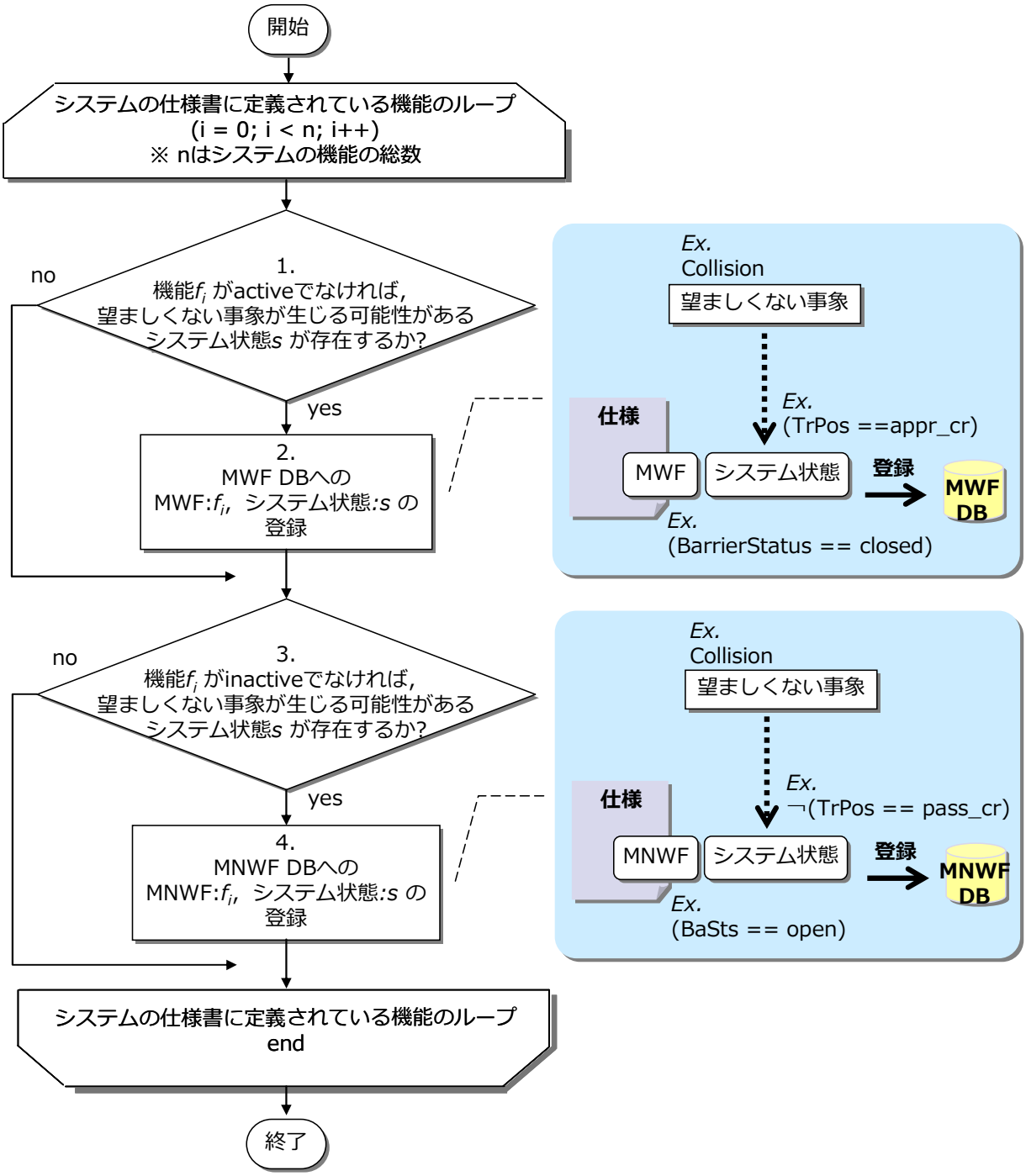


図 4-6　MWF，MNWF 識別と DB 登録のプロセス

望ましくない事象	MWF	システム状態	MWF非起動に関する事象
Collision	(BaSts == closed)	(TrPos == appr_cr)	(TrPos == appr_cr)∧ ¬(BaSts == closed)
	…	…	…
	…	…	…
(TrPos == appr_cr)∧ ¬(BaSts == closed)	(TrSndMsg == act_order)	(TrDetPos == appr_cr)	(TrDetPos == appr_cr)∧ ¬(TrSndMsg == act_order)
…	…	…	…

MWF DBの関数：
getMWFdeact()
実行時に生成

図 4-7　MWF DB の構造

望ましくない事象	MNWF	システム状態	MNWF起動に関する事象
Collision	(BaSts == open)	¬(TrPos == pass_cr)	¬(TrPos == pass_cr)∧ (BaSts == open)
	…	…	…
¬(TrPos == pass_cr)∧ (BaSts == open)	(SeSndMsg == pass_rpt)	¬(SeDetTrPos == pass_cr)	¬(SeDetTrPos == pass_cr)∧ (SeSndMsg == pass_rpt)
…	…	…	…

MNWF DBの関数：
getMNWFact()
実行時に生成

図 4-8　MNWF DB の構造

4.3.4 書換えルールによる望ましくない事象の具体化

望ましくない事象の具体化は, 項書換え系の考えに基づく知識の書換えにより形式化されている．提案手法における知識の書換えルールは，以下に示す 2 つである．

- ルール 1 :「MWF 非起動」の観点による具体化
- ルール 2 :「MNWF 起動」の観点による具体化

書換えルールを適用可能な知識「望ましくない事象」が存在する場合，書換えルールの 1 つによって知識の書換えが行われる．図 4-9 に，ルール 1 :「MWF 非起動」の観点による具体化を示す．図 4-9 は，項書換え系の考えを基に，知識の書換えルールを図示している．図 4-9 のボックスは，知識を表す．ルール適用の前提条件における関数「existUndEvtMWFDB()」は，引数として与えられた知識「望ましくない事象」が，MWF DB の項「望ましくない事象」に登録されているか否かを判定する関数である．知識「MWFdeactFlag」は，知識「望ましくない事象」に対して，ルール 1 の知識の書換えが適用された際に，知識「望ましくない事象」に対して設定されるフラグである．知識「MWFdeactFlag」は，同一の知識「望ましくない事象」に対して，ルール 1 の知識の書換えが複数回適用されることを防ぐ制御情報として用いられる．知識の書換えは，書換えルールに与えられた知識「望ましくない事象」が MWF DB に登録されていて，かつ，知識「望ましくない事象」に対して，ルール 1 の知識の書換えが未だ行われていない場合に適用される．ルール 1 の知識の書換え時に，関数「setMWFdeactFlg()」および関数「getMWFdeact()」が実行される．関数「setMWFdeactFlg()」が実行されることで, 知識「望ましくない事象」に対して, 知識「MWFdeactFlag」が設定される．関数「getMWFdeact()」が実行されることで，引数として与えられた知識「望ましくない事象」に対応する知識「MWF 非起動に関する事象」のリストが MWF DB から取得される．知識「MWF 非起動に関する事象」は，本関数の実行時に，MWF DB に登録されている知識「望ましくない事象」に対応する知識「MWF」および知識「システム状態」を基に生成される．その際，知識「MWF 非起動に関する事象」は，4.3.2 の(3)に従い，生成される．生成された知識「MWF 非起動に関する事象」は，MWF DB に登録される．

ルール 1 について，図 4-9 の無線踏切システムの具体例を用いて説明する．知

識「Collision」が，MWF DB に登録されていて，かつ，知識「Collision」に対して，知識「MWFdeactFlag」が設定されていない．つまり，望ましくない事象に対する MWF が存在し，かつ，ルール 1 による知識の書換えが行われていない．そのような場合に，知識「Collision」が，知識「Collision」および知識「(TrPos == appr_cr)∧¬(BaSts == closed)」らに書き換えられている．図 4-9 に示す具体例は，図 4-6 および図 4-7 に示す無線踏切システムの具体例に対応している．

図 4-10 に，ルール 2 :「MNWF 起動」の観点による具体化を示す．ルール 2 の考え方は，ルール 1 と同様である．

4.3.1 で述べたとおり，ルール 1 による知識の書換えの後に，「MWF 非起動に関する事象」は，「望ましくない事象」に置き換わる．同様に，ルール 2 による知識の書換えの後に，「MNWF 起動に関する事象」は，「望ましくない事象」に置き換わる．これらの「望ましくない事象」が，MWF DB もしくは MNWF DB に存在する場合，「望ましくない事象」に対して，再び知識の書換えルールが適用される．

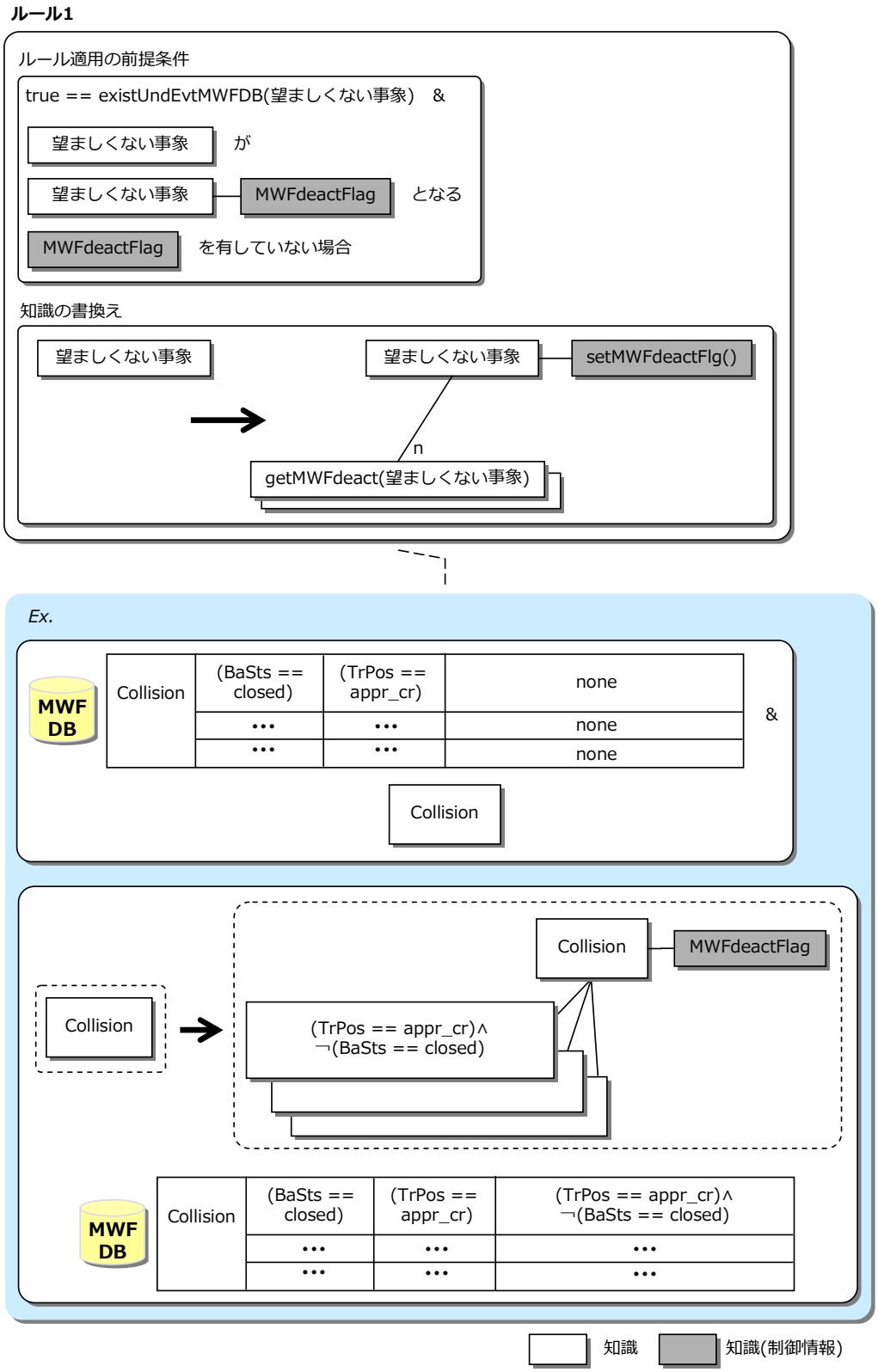


図 4-9　ルール 1：「MWF 非起動」の観点による具体化

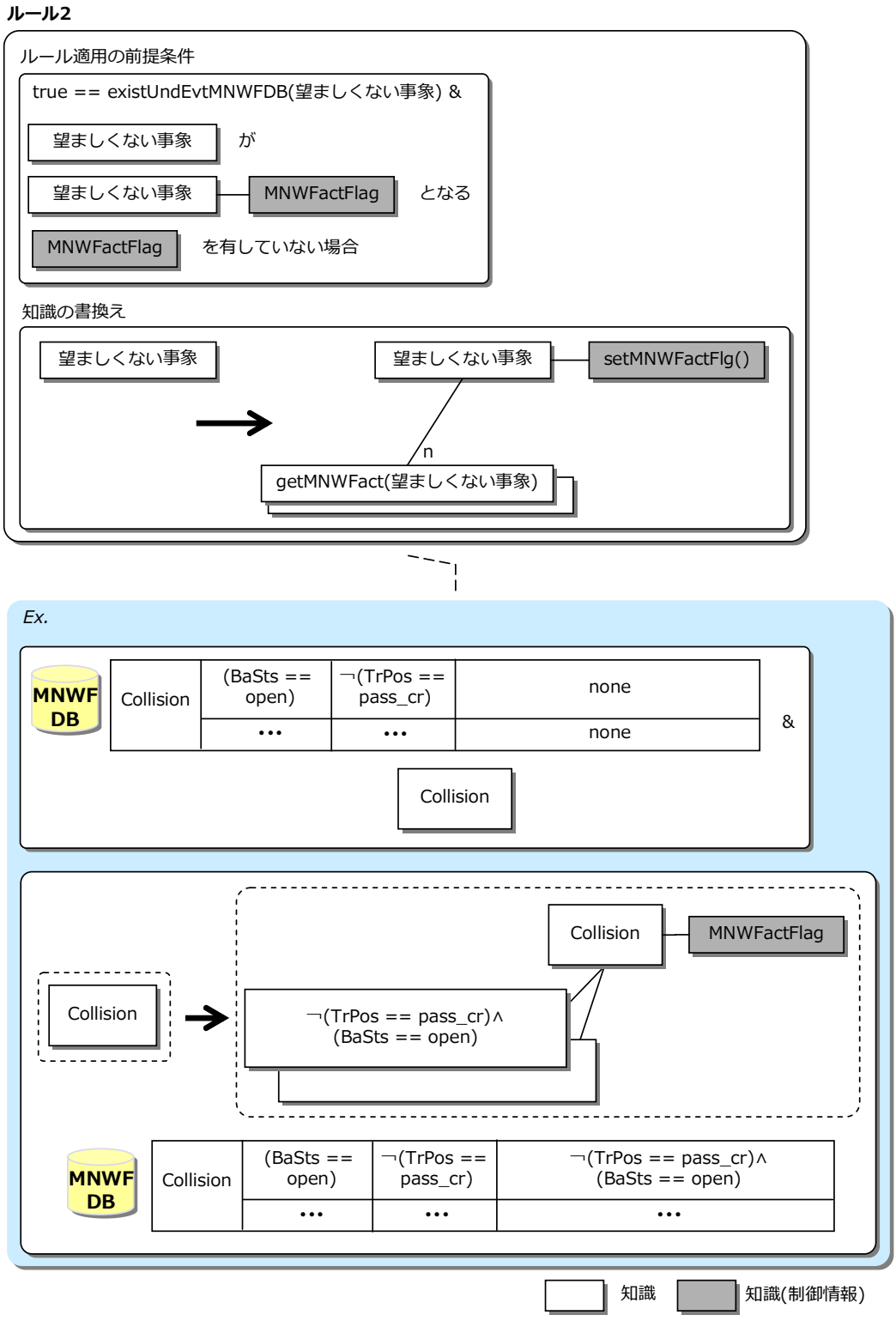


図 4-10　ルール 2：「MNWF 起動」の観点による具体化

4.3.5　安全性プロパティの生成

「MWF 非起動」および「MNWF 起動」の観点で具体化された望ましくない事象を基に，具体的な安全性プロパティが生成される．安全性プロパティは，「システムにとって望ましくない事象が決して起こらない」性質である．従って，具体化された望ましくない事象の否定（¬）が取られる．否定化された事象に対して，安全性プロパティに関する時相論理演算子が付与される．CTL により安全性プロパティが表現される場合には，否定化された事象に対して AG が付与される．LTL により安全性プロパティが表現される場合には，否定化された事象に対して G が付与される．無線踏切システムを例にいえば，CTL で安全性プロパティが表現される場合，具体化された望ましくない事象「(TrPos == appr_cr)∧¬(BaSts == closed)」を基に，安全性プロパティとして「AG ¬((TrPos == appr_cr)∧¬(BaSts == closed))」が生成される．

4.4　評価実験

提案手法の評価実験を行い，提案手法の有効性を確認する．まず，無線踏切システム[65][66]に対して，被験者は，経験やスキルに基づく従来の方法によりモデル検査における具体的な安全性プロパティを導出する．つぎに，被験者は，提案手法を修得し，提案手法により具体的な安全性プロパティを導出する．従来方法および提案手法により導出された具体的な安全性プロパティを比較し，従来方法では導出されなかった安全性プロパティが，提案手法により導出されることを確認する．また，従来方法により具体化された安全性プロパティが，提案手法によってさらに具体化されることを確認する．

4.4.1　適用事例

評価実験の事例として，無線踏切システムを選定した．無線踏切システムを選定した理由は，本システムがドイツ鉄道の実際の開発事例に基づく，ソフトウェアによりシステムの制御を行う複雑なシステムだからである．図 4-11 に，無線踏切システムの概要を示す．列車が踏切に接近する際，踏切に対して無線によるメッセージが送信される．踏切には，自動車や歩行者などのロードユーザ

用の遮断機，信号灯が設置されている．ロードユーザは，遮断機が閉じている場合，信号灯が点灯している場合は，踏切の前で待機する．踏切には，列車の踏切通過を検知するセンサが設置されている．

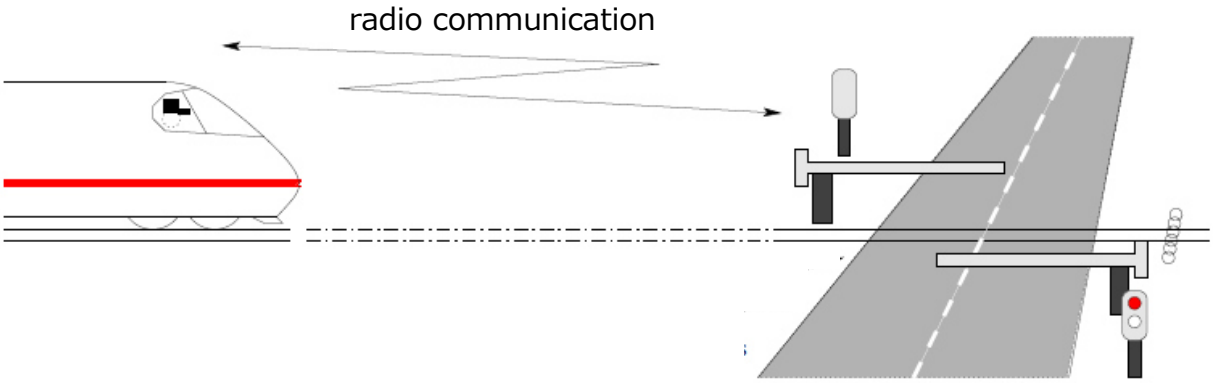


図 4-11　無線踏切システム

4.4.2　評価実験の被験者

評価実験の被験者として，4 人の技術者を選定した．表 4-2 に，評価実験の被験者を示す．評価実験の被験者を A，B，C，D とする．被験者の計算機システムに関する業務経験年数は，それぞれ，15 年，13 年，11 年，4 年である．それぞれの被験者は，モデル検査の適用経験を有する．また，被験者 A および被験者 B は，評価実験の適用事例である鉄道に関するドメイン知識を有している．被験者 C および被験者 D は，特別なドメイン知識を有していない．

表 4-2　評価実験の被験者

	計算機システムの業務経験年数	モデル検査の実践経験	ドメイン知識
被験者A	15年	有	有
被験者B	13年	有	有
被験者C	11年	有	無
被験者D	4年	有	無

4.4.3 具体的な安全性プロパティの導出

モデル検査における具体的な安全性プロパティの導出にあたり，被験者は，文献[66]の仕様を基に，無線踏切システムを理解した．従来方法および提案手法により，具体的な安全性プロパティを導出する過程においても，文献[66]の仕様を参照した．評価実験では，無線踏切システムにおける非形式的な望ましくない事象として，「Collision」が与えられた．被験者は，「Collision」を入力として，従来方法および提案手法を適用した．従来方法および提案手法による具体的な安全性プロパティの導出結果をつぎに示す．なお，評価実験では，CTL を用いて具体的な安全性プロパティを表現した．また，従来方法による安全性プロパティの導出後に，提案手法による導出を実施したものの，それらの実施には一定の期間をおいた．したがって，無線踏切システムに対する理解度という観点において，従来方法の実施後に提案手法を実施した影響は小さいと考える．

(a) 従来方法

それぞれの被験者は，計算機システムに関する業務経験，個人に蓄積されたモデル検査の要点などを基に，「Collision」に対する具体的な安全性プロパティを導出した．図 4-12 に，導出結果の代表例として，被験者 A が従来方法により導出した具体的な安全性プロパティを示す．図 4-12 において，ボックスは事象を示す．縞網掛け丸は，従来方法により導出された具体的な安全性プロパティを示す．[ ]内の ID は，導出された安全性プロパティに対して一意に付与される番号である．

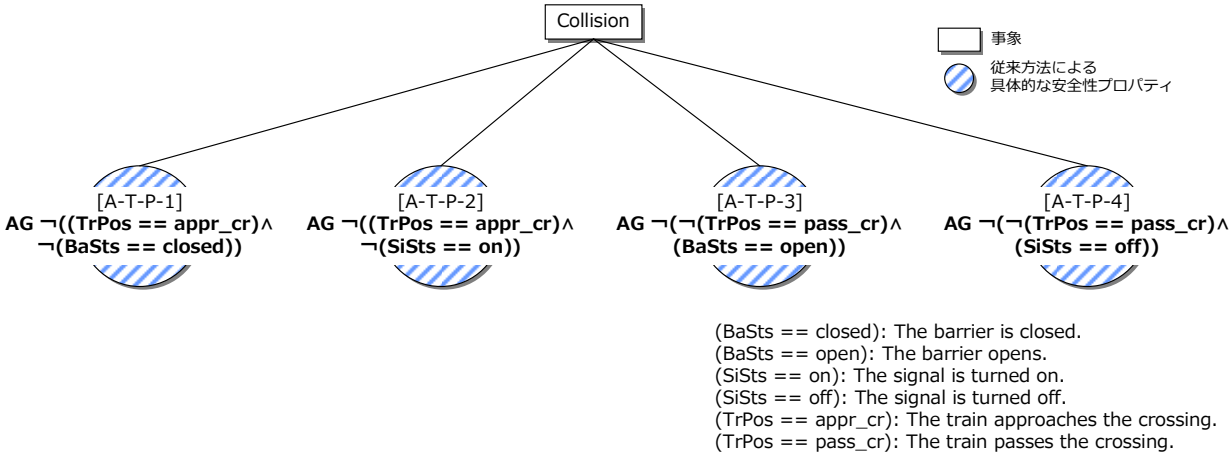


図 4-12　従来方法による具体的な安全性プロパティの導出結果（被験者 A）

(b) 提案手法

従来方法による具体的な安全性プロパティの導出後，被験者は，提案手法を修得した．その後，それぞれの被験者は，提案手法を用いて，「Collision」に対する具体的な安全性プロパティを導出した．図 4-13 に，導出結果の代表例として，被験者 A が提案手法により導出した具体的な安全性プロパティを示す．図 4-13 において，ボックスは事象を示す．網掛け丸は，提案方法により導出された具体的な安全性プロパティを示す．[ ]内の ID は，導出された事象および安全性プロパティに対して一意に付与される番号である．事象の間に記述される「MWF 非起動」および「MNWF 起動」は，具体的な安全性プロパティが導出される過程において，導出された事象が，「MWF 非起動」または「MNWF 起動」のどちらの観点による事象であるかを表す．図 4-13 の見方について，A-P-E-3 の事象は，「Collision」から，「MWF 非起動」の観点により導出されていることを表す．A-P-E-3-2 の事象は，A-P-E-3 の事象から，「MNWF 起動」の観点により導出されていることを表す．A-P-E-2-1 のボックスにおける（A-P-E-1-1）は，A-P-E-2-1 と A-P-E-1-1 が同じ事象であることを指す．A-P-E-5-1 のボックスにおける（A-P-E-4-1）も，A-P-E-5-1 と A-P-E-4-1 が同じ事象であることを指す．点線で囲まれた事象は，「MWF 非起動」および「MNWF 起動」の観点で具体化された最終的な事象である．

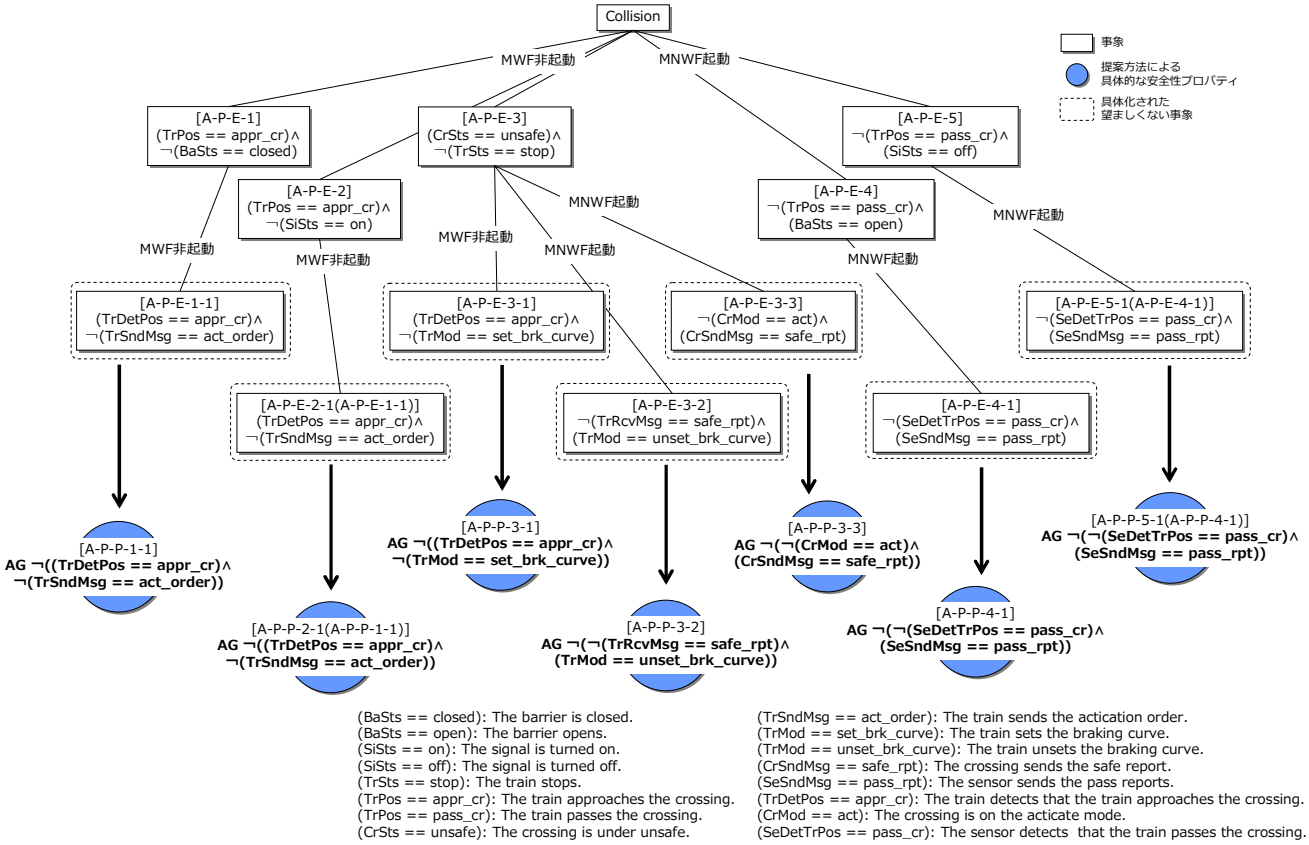


図 4-13　提案手法による具体的な安全性プロパティの導出結果（被験者 A）

4.4.4 実験結果

それぞれの被験者における実験結果の代表例として，被験者 A が従来方法により導出した安全性プロパティ（図 4-12）と，提案手法により導出した安全性プロパティ（図 4-13）の比較結果を図 4-14 に示す．図 4-14 において，ボックスは事象を示す．ボックス内に記述される ID は，図 4-12 および図 4-13 に示される，それぞれの事象に付与された ID である．縞網掛け丸は，従来方法により導出された具体的な安全性プロパティを示す．網掛け丸は，提案方法により導出された具体的な安全性プロパティを示す．提案手法により導出された A-P-E-1 は，否定化され安全性プロパティに関する時相論理演算子が付与されると，従来方法により導出された A-T-P-1 になる．従って，A-P-E-1 を A-T-P-1 と対応付けている．A-P-E-2 と A-T-P-2，A-P-E-4 と A-T-P-3，A-P-E-5 と A-T-P-4 についても同様である．評価実験の結果，従来方法では導出されなかった A-P-P-3-1，A-P-P-3-2，A-P-P-3-3 の安全性プロパティが，提案手法により導出されている．この結果から，提案手法によって，安全性プロパティの網羅性が向上することを確認した. また, 提案手法により導出された A-P-P-1-1, A-P-P-2-1（A-P-P-1-1），A-P-P-4-1，A-P-P-5-1（A-P-P-4-1）の具体的な安全性プロパティは，従来方法により導出された A-T-P-1，A-T-P-2，A-T-P-3，A-T-P-4 の安全性プロパティを，それぞれ具体化した関係を持つ．この結果から，提案手法によって，安全性プロパティの具体性が向上することを確認した．

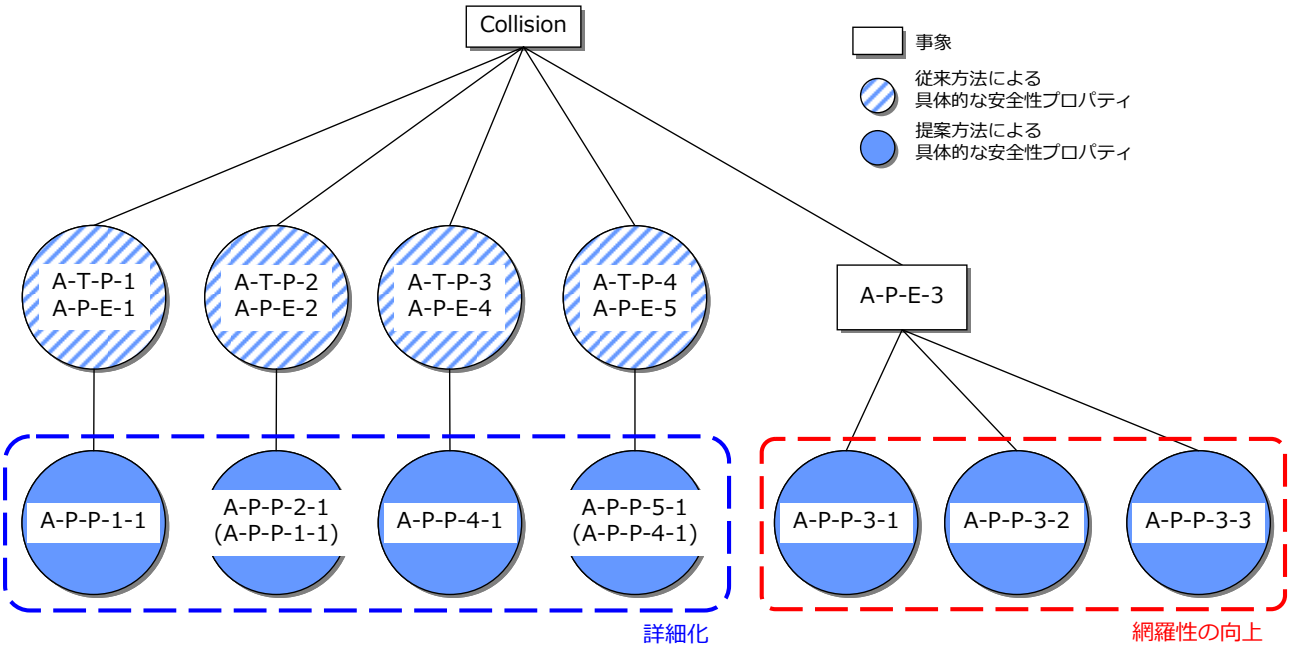


図 4-14　導出された具体的な安全性プロパティの比較（被験者 A）

提案手法による具体的な安全性プロパティの網羅性および具体性に着目した，それぞれの被験者による実験結果を表 4-3 に示す．

表 4-3 の項：「従来方法による安全性プロパティ」は，従来方法により導出された具体的な安全性プロパティの個数を表す．項：「提案手法により新たに導出された安全性プロパティ」は，従来方法により導出された具体的な安全性プロパティに対し，提案手法により新たに導出された具体的な安全性プロパティの個数を表す．安全性プロパティに関する網羅性の評価では，網羅的な安全性プロパティの集合と，提案手法により導出された安全性プロパティを比較する方法が考えられる．しかし，網羅的な安全性プロパティの集合を定義することは困難である．理由は，「MNWF 起動」に関する安全性プロパティは，開集合である可能性があるためである．したがって，本評価実験では，安全性プロパティの網羅性を評価する尺度として，提案手法により新たに導出された具体的な安全性プロパティの個数を採用する．この個数がプラスの場合，提案手法によって，具体的な安全性プロパティの網羅性が向上していることを意味する．図 4-14 の場合，A-P-P-3-1，A-P-P-3-2，A-P-P-3-3 の 3 つの事象に相当する．表 4-3 の項：「従来方法により具体化された安全性プロパティに対する，提案手法により具体化された安全性プロパティ」は，従来方法により具体化された安全性プロパティに対し，提案手法により，さらに具体化された安全性プロパティの個数を表す．安全性プロパティに関する具体性の評価では，安全性プロパティの具体性の十分さを定義することは困難である．理由は，安全性プロパティの具体性の十分さは，モデル検査におけるモデルの具体性に依存する．場合によっては，導出された安全性プロパティから，そのモデル自体を具体化することがあり，安全性プロパティの具体性の基準が定まらない可能性があるためである．したがって，本評価実験では，安全性プロパティの具体性を評価する尺度として，提案手法によりさらに具体化された安全性プロパティの個数を採用する．この個数がプラスの場合，提案手法によって，安全性プロパティの具体性が向上していることを意味する．図 4-14 の場合，A-P-P-1-1 および A-P-P-4-1 の 2 つの事象に相当する．A-P-P-2-1 および A-P-P-5-1 の事象は，それぞれ A-P-P-1-1 および A-P-P-4-1 と同じであるため，表 4-3 では計上していない．表 4-3 の項：「網羅性の向上率」は，従来方法により導出された具体的な安全性プロパティに対し，提案手法により新たに導出された具体的な安全性プロパティの相対的な向上率を表す．被験者 A を例にいえば，網羅性の向上率 75%は，従来方法により導出

された安全性プロパティの個数 4，および，提案手法により新たに導出された具体的な安全性プロパティの個数 3 を基に算出されている．表 4-3 の項 :「具体性の向上率」は，従来方法により具体化された安全性プロパティに対し，提案手法により，さらに具体化された安全性プロパティの相対的な向上率を表す．被験者 A を例にいえば，具体性の向上率 50%は，従来方法により具体化された安全性プロパティの個数 4，および，提案手法により，さらに具体化された安全性プロパティの個数 2 を基に算出されている．

評価実験の結果から，被験者 A，B，C において，提案手法によって，具体的な安全性プロパティの網羅性が向上していることを確認した．また，すべての被験者において，提案手法によって，安全性プロパティの具体性が向上していることを確認した．

表 4-3　網羅性および具体性に着目した実験結果

被験者	従来方法による安全性プロパティ	提案手法により新たに導出された安全性プロパティ	従来方法により具体化された安全性プロパティに対する，提案手法により具体化された安全性プロパティ	網羅性の向上率	具体性の向上率
A	4	3	2	75%	50%
B	6	3	3	50%	50%
C	5	3	3	60%	60%
D	4	0	3	0%	75%

4.4.5　導出されたプロパティによるモデル検査の実施

提案手法により導出された具体的な安全性プロパティを基に, 無線踏切システムに対してモデル検査が実施された．モデル検査を実施するにあたり，モデル検査ツールとして，CTL によりプロパティを表現する UPPAAL が採用された．無線踏切システムの UPPAAL モデル，それぞれの被験者において提案手法により導出された具体的な安全性プロパティを用いて，モデル検査が実施された．その結果，提案手法により導出された安全性プロパティによって，無線踏切システムの検証が可能であることを確認した．

4.5　考察

4.4 で示した評価実験の結果を基に，提案手法の有効性を述べる．また，提案手法の課題を述べる．

4.5.1　提案手法の有効性

評価実験の結果，被験者 A，B，C において，提案手法によって具体的な安全性プロパティの網羅性が向上することを確認した．それについて，被験者 A による導出結果を基に考察する. 図 4-14 において, A-P-P-3-1, A-P-P-3-2, A-P-P-3-3 の具体的な安全性プロパティは，従来方法では導出されていない．提案手法では，「MWF 非起動」および「MNWF 起動」の 2 つの観点を用いた系統的な方法によって，システムの望ましくない事象が具体化される．このことが，提案手法において，具体的な安全性プロパティの網羅性向上に貢献していると考える．

今回の評価実験において，被験者 D については，具体的な安全性プロパティの導出において, 網羅性の向上を確認することができなかった. それについて, 被験者 D による導出結果を基に考察する．図 4-15 に，被験者 D が従来方法および提案手法により導出した具体的な安全性プロパティの比較結果を示す．図 4-15 の見方は，図 4-14 と同様である．図 4-15 に示すとおり，従来方法により導出された具体的な安全性プロパティ（D-T-P-1）と，提案手法により導出された事象（D-P-E-1）が対応関係にある．D-T-P-2 および D-P-E-2 も，それと同様である．また，従来方法により導出された具体的な安全性プロパティ（D-T-P-3）と，提案手法により導出された具体的な安全性プロパティ（D-P-P-3）が同じ結

果になっている．D-T-P-4 および D-P-P-1-1 も，それと同様である．これは，被験者 D が，経験やスキルに基づく従来の方法として，提案手法で採用される「MWF 非起動」および「MNWF 起動」の観点を，潜在的に有していた可能性が考えられる．

また，評価実験の結果，すべての被験者において，提案手法によって安全性プロパティの具体性が向上することを確認した．それについて，被験者 A による導出結果を基に考察する．図 4-14 において，従来方法により導出された具体的な安全性プロパティ（A-T-P-1）と，提案手法により導出された事象（A-P-E-1）は，対応関係にある．その事象を基に，提案手法では，「MWF 非起動」の観点により，A-P-P-1-1 が導出されている．同様に，A-T-P-3/A-P-E-4 の事象を基に，「MNWF 起動」の観点により，A-P-P-4-1 が導出されている．提案手法では，3.3 に示す，システムの仕様書から MWF および MNWF が識別されるプロセスが繰り返される．このことが，提案手法において，安全性プロパティの適切な具体化に貢献していると考える．

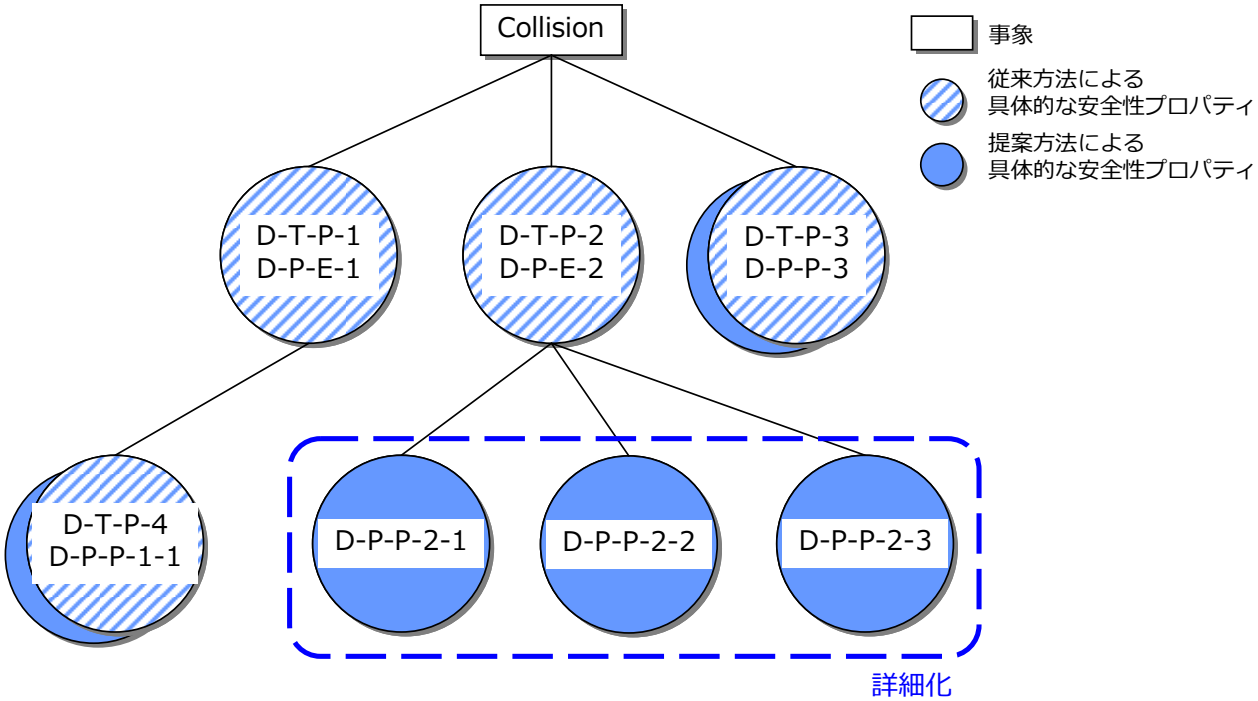


図 4-15　導出された具体的な安全性プロパティの比較（被験者 D）

4.5.2　提案手法の課題

4.5.1 において，提案手法の有効性を確認することができた．本項では，提案手法に関する今後の課題を述べる．

システムの望ましくない事象を系統的に識別する方法を確立する必要がある．提案手法は，モデル検査の検証対象となるシステムの望ましくない事象を入力としている．望ましくない事象は，それが発現する前に識別することが困難な場合が多い．また，発現して初めて望ましくない状態として認識されることも多い．本課題の対策としては，望ましくない事象，および，その事象に対して提案手法により作成される知識ネットワーク（図 4-4 に示す，望ましくない事象，「MWF 非起動に関する事象」および「MNWF 起動に関する事象」の関連図）を蓄積することで，望ましくない事象の識別に関する網羅性を向上させる方法が考えられる．望ましくない事象を識別する際に，蓄積された望ましくない事象および知識ネットワークを参照する．それにより，望ましくない事象の識別抜けを少なくできる可能性がある．また，提案手法により作成される知識ネットワークの分析から，新たな望ましくない事象が発見できる可能性もある．

「MWF 非起動」および「MNWF 起動」の観点以外にも，導出する安全性プロパティの網羅性を向上させる観点が存在する可能性がある．評価実験において，「MWF 非起動」および「MNWF 起動」の観点の適用は，導出する安全性プロパティの網羅性の向上に貢献することが確かめられた．「MWF 非起動」および「MNWF 起動」の観点は，意図しない機能の非起動および起動に着目している．これらが望ましくない事象を引き起こす可能性があることはいうまでもない．ただし，機能の異常についても，望ましくない事象を引き起こす可能性がある．本課題の対策としては，安全性プロパティの導出において，HAZOP[76]のガイドワードを用いる方法が考えられる．機能の出力値が大きすぎる／小さすぎる，機能の動作が早すぎる／遅すぎるなどのガイドワードを用いる．それにより，導出する安全性プロパティの網羅性が，さらに向上する可能性がある．

提案手法の属人性を，さらに低減させる必要がある．提案手法では，モデル検査の検証対象となるシステムの仕様書から，MWF および MNWF を識別する．その作業には，実施者の属人性が関与する可能性がある．特に，仕様書から MNWF を識別する作業は，実施者の経験やスキルに大きく依存する可能性がある．本課題の対策としては，提案手法により作成される知識ネットワークを蓄積しておく．蓄積された知識ネットワークにおける「MWF 非起動」および

「MNWF 起動」に関する事象を参考に，MWF および MNWF を識別する方法が考えられる．また，MWF および MNWF の識別時に，それらの識別をガイダンスする方法が考えられる．[77]では，事故の根本原因分析時の属人性を排除するために，システムズエンジニアリングにおける Dual Vee Model[78]および ISO9000 品質マネジメントモデル[79]を採用している．これらのモデルをガイダンスとして用いることで，根本原因分析時の属人性の排除を図っている．仕様書から MWF および MNWF を識別する際に, ガイダンスを用いる. それにより, MWF および MNWF を識別する際の属人性の低減が期待できる.

4.6 まとめ

本章では, モデル検査における具体的な安全性プロパティを導出する手法を提案した．提案手法では，「MWF 非起動」および「MNWF 起動」の 2 つの観点に基づき，システムの望ましくない事象を具体化する．それにより，システムの望ましくない事象から，具体的な安全性プロパティを導出する．提案手法の評価実験として，無線踏切システムに対して提案手法を適用した．その結果，従来の方法では導出されなかった具体的な安全性プロパティが，提案手法により導出されることを確認した．また，従来の方法で具体化された安全性プロパティが，提案手法により，さらに具体化されることを確認した．評価実験の結果から，提案手法は，モデル検査における具体的な安全性プロパティの導出において，導出する安全性プロパティの網羅性および具体性の向上を実現する手法であることが確かめられた.

第5章　全体考察

本章では，本論文における提案手法の成果を考察する．また，提案手法に関する今後の発展を述べる．

5.1　提案手法の成果

本論文では，図 2-13 に示した，モデル検査における工程 2 および工程 3 の課題の解決を目的として，複雑な組込みシステムに対してモデル検査を適用する手法を提案した．提案手法では，モデル検査の検査対象であるシステムの望ましくない事象から，具体的な安全性プロパティを導出する．そして，導出した具体的な安全性プロパティを基に，構成要素および構成要素間の連携部分をそれぞれ個別にモデル検査する．本論文において，提案手法は，工程 2 および工程 3 の課題に対して有効であることを示した．これらの結果から，複雑な組込みシステムに対するモデル検査適用のハードルを下げる手法を確立することができたと考える．

また，本論文では，モデル検査の適用者の視点から見た課題に着目し，その課題解決を試みた．産業界における検証技術として，モデル検査をさらに浸透させるにあたり，モデル検査の適用者の視点での課題解決は，重要な取り組みである．その点においても，本論文の価値は高いと考える．

5.2　今後の発展

前節において, 提案手法の成果を述べた. 本節では, 提案手法の成果を踏まえ, 本研究の今後の発展性を述べる．

5.2.1　複雑なシステムに対する更なる適用性の向上

提案手法では, システムの構成要素および構成要素の連携部分に着目してモデル検査を実施する．構成要素において複数の処理が並行動作するシステムの場合，構成要素のモデル検査において，状態爆発が発生する可能性は否定できない．その場合，構成要素に対してアーキテクチャ設計を実施することにより，構成要素をさらに分割する．分割したそれらに対して提案手法を適用すること

で，状態爆発を回避する方法が考えられる．また，多くの構成要素が複雑に連携し合うシステムの場合についても，構成要素の連携部分のモデル検査において，状態爆発が発生する可能性は否定できない．その場合，連携部分を分割し，分割したそれらに対して，Cobleigh らが提案した Assume-Guarantee Reasoning[62]を適用する．それにより，状態爆発を回避する方法が考えられる．

5.2.2 包括的なプロパティ導出手法の確立

本論文では，安全性プロパティの導出ルールを定めた．モデル検査におけるプロパティには，安全性プロパティの他に活性プロパティ[5]がある．活性プロパティとは，システムにとって望ましい事象がいつかは起こる性質である．システムにおける望ましい事象とは，システムに求めるサービスを，システムが確実に提供することである．それは，モデル検査の対象となるシステムの，上位の仕様書に定義されている要件や機能の場合が多い．モデル検査の対象および上位の仕様書との対応関係は，トレーサビリティマトリクスにまとめられている．トレーサビリティマトリクスを用いることで，活性プロパティの導出ルールを定義できる可能性がある．その場合，モデル検査における包括的なプロパティ導出手法の確立が期待できる．

5.2.3 属人性を排除した FTA の実現

FTA[70][71][72]では，Fault Tree の作成を通した演繹的なアプローチによって，望ましくない事象の原因となる事象を識別する．FTA には，Fault Tree における親事象から子事象の導出が，実施者の経験やスキルに依存するという課題がある．FTA での事象の導出における属人性の排除について，提案手法で採用した「MWF 非起動」および「MNWF 起動」の観点が有効である可能性がある．MWF は，「その意図しない非起動が，システムの望ましくない事象を引き起こす可能性がある機能」である．MNWF は，「その意図しない起動が，システムの望ましくない事象を引き起こす可能性がある機能」である．機能に着目した FTA を実施する場合，「機能の喪失」および「機能の異常」という観点を用いて事象を導出する方法が考えられる．「機能の喪失」に関する事象を導出する際に，「MWF

非起動」の観点を用いる．「機能の異常」に関する事象を導出する際に，「MNWF起動」の観点を用いる．そして，機能の出力値が大きすぎる／小さすぎる，機能の動作が早すぎる／遅すぎる，などの HAZOP[76]のガイドワードを用いて，「機能の異常」に関する事象を補う．Fault Tree の作成において，頂上事象の原因となる事象を導出する際の観点を定義する．それにより，属人性を極力排除した，より網羅性の高い FTA の実施が期待できる．

5.2.4 組込みシステム以外の複雑なシステムへの適用

本論文では，複雑な組込みシステムの信頼性向上を目的として，提案手法を示した．提案手法は，組込みシステムのみならず，情報系システムなどの，産業界における他の複雑なシステムに対しても適用できる可能性がある．ただし，組込みシステムと情報系システムの大きな違いは，システムの規模である．情報系システムは，構成要素が多いことが想定される．また，それに伴い，構成要素の連携部分が多いことが想定される．構成要素および構成要素の連携部分の多さは，提案手法を適用する際のコストの増加に繋がる．情報系システムに対して提案手法を適用する際には，状態遷移モデルの作成を含めた提案手法の自動化が鍵となる．提案手法を可能な限り形式化し，計算機の支援を取り入れる．それにより，大規模なシステムに対する，コストを抑えた提案手法の適用が期待できる．

5.2.5 技術系システム以外への提案手法の適用

モデル検査は, 従来は, 技術システムの状態遷移モデルを検証対象としてきた．しかし，状態遷移を有するシステムは，技術システムだけではない．近年，技術システム以外へのモデル検査の適用が試みられている．[80]では，ロールプレイングゲームのシナリオを基に，状態遷移モデルを作成している．そして，ゲームにおけるすべてのイベントにたどり着けること，エンディングにたどり着けないことがないことを，モデル検査で確認している．また，[12]では，国際宇宙ステーションにおける異常発生時の運用手順を基に，状態遷移モデルを作成している．そして，異常時の運用手順を用いて，システム異常に対応する故障

原因に確実にたどり着けることを，モデル検査で確認している．これらの試みを踏まえると，モデル検査は，法律や規則などに対しても適用できる可能性がある．技術システム以外へのモデル検査の適用，そのような検証対象における状態爆発の防止やプロパティの導出は，提案手法の今後を展望するに当たり，興味深い研究分野である．

第6章　結論

本論文では，複雑な組込みシステムに対して，モデル検査を適用する手法を提案した．複雑なシステムに対してモデル検査を適用する際には，つぎに示す課題 1 および課題 2 が存在する．

- 課題 1 : モデル検査において状態爆発が発生する
- 課題 2 : 具体的な安全性プロパティの導出において，プロパティの見逃しが発生することや具体化が不十分になる

課題 1 に対して, システムの構成要素および構成要素の連携に関連する部分を, それぞれ個別にモデル検査する手法を提案した．また，課題 2 に対して，「MWF 非起動」および「MNWF 起動」の観点を用いて，具体的な安全性プロパティを導出する手法を提案した．課題 1 および課題 2 に対する提案手法の評価実験において，それぞれの手法の有効性が確かめられた．本論文の提案手法により，モデル検査の研究領域の発展および産業界における複雑な組込みシステムの信頼性向上に貢献することができた．

本研究の今後の方向性は，つぎのとおりである．課題 1 に対する提案手法について，手法を形式化し，手法の健全性を厳密に示す．また，形式化された仕様および形式化されたトレーサビリティマトリクスを用いることで，連携部分における状態遷移モデルの正しさを確保する方法を検討する．課題 2 に対する提案手法について，望ましくない事象の系統的な識別方法を検討する．また，安全性プロパティの導出における，「MWF 非起動」および「MNWF 起動」以外の有効な観点を検討する．これらにより，複雑なシステムに対する，より網羅的なモデル検査手法の確立を図る．また，仕様書からの MNWF の識別には，属人性が入り込む可能性がある．MNWF の識別において，属人性の関与を低減するような仕組みを検討する．また，提案手法には，大きな発展性がある．特に，複雑なシステムに対する，さらなる適用性の向上，ならびに，包括的なプロパティ導出手法の構築に着目し，その実現を目指す．

参考文献

[1] Leveson, G. N.: SAFEWARE: SYSTEM SAFETY AND COMPUTERS, pp.515-553, Addison-Wesley (1995).

[2] 清水久二:アリアン 5 の爆発事故とソフトウェア安全性に関する国際規格, 安全工学会安全工学誌，Vol.41，No.1，pp.39-42（2002）.

[3] Clarke, M. E. and Wing, M. J.: Formal methods: State of the art and future directions, ACM Computing Surveys, Vol.28, No.4, pp.626-643 (1996).

[4] Clarke, M. E., Grumberg, O. and Long, E. D.: Model Checking and abstraction, ACM Transactions on Programming Languages and Systems, Vol.16, No.5, pp.1512-1542 (1994).

[5] Alpern, B. and Schneider, B. F.: Defining liveness, Information Processing Letters, Vol.21, No.4, pp.181-185(1985).

[6] International Council on Systems Engineering (INCOSE): INCOSE SYSTEMS ENGINEERING HANDBOOK version 3.1, 1.5 of 6, INCOSE, USA (2007).

[7] Institute of Electrical and Electronics Engineers (IEEE): IEEE Standard for System and software engineering – System life cycle processes, IEEE 15288-2008 (2008).

[8] American National Standard Institute (ANSI)/Electronic Industries Alliance (EIA): ANSI/EIA Standard for Process for Engineering a System, ANSI/EIA 632-1999 (1999).

[9] Institute of Electrical and Electronics Engineers (IEEE): IEEE Standard for Application and Management of the Systems Engineering Process, IEEE 1220-2005 (2005).

[10] The National Aeronautics and Space Administration: Computer-Based Control System Safety Requirements, SSP 50038 Revision B (1995).

[11] Basili, R., Selby, R.: “Comparing the effectiveness of software testing strategies”, IEEE Transactions on Software Engineering, Vol.SE-13, No.12, pp.1278-1296 (1987).

[12] 加藤淳，神武直彦，片平真史：宇宙機ソフトウェアに対する IV&V とその適用事例，情報処理学会 組込みシステムシンポジウム 2007，IPSJ Symposium Series Vol.2007，No.8，pp.120-129（2007）.

[13] Rompaey, K. V., Verkest, D, Bolsens, I. and Man, H. D.: CoWare – A design environment for heterogeneous hardware/software systems, Proceedings of the European Design Automation Conference, pp.252-257 (1996).

[14] Tassey, G.: The Economic Impacts of Inadequate Infrastructure for Software Testing, National Institute of Standards and Technology, RTI Project Number 7007.011 (2002).

[15] 独立行政法人情報処理推進機構：形式手法活用ガイド 導入の手引き（2012）.

[16] Spivey, J. M.: Z Notation -A Reference Manual, Longman Higher Education (1988).

[17] Jones, C. B.: Systematic Software Development Using VDM, Prentice Hall (1990).

[18] Abrial J. R., Lee, M. K. O., Neilson, D. S., Scharbach, P. N. and Sorensen, I. H.: The B-method, Proceedings of the 4th International Symposium of VDM Europe on Formal Software Development, Vol.2, pp. 98-405 (1991).

[19] Abrial J. R.: Modeling in Event-B: System and Software Engineering, Cambridge University Press (2010).

[20] Goguen, J. A. and Winkler, T.: Introducing OBJ3, Tech. Rep. SRI-CSL-88-09 (1988).

[21] Diaconescu, R. and Futatsugi, K.: Logical Foundations of CafeOBJ, Theoretical Computer Science, Vol.285, pp. 289-318 (2002).

[22] Bolognesi, T. and Brinksma, E.: Introduction to the IS0 specification language LOTOS, Computer Networks and ISDN Systems, Vol.14, pp.25-59 (1987).

[23] Hoare, C. A. R.: Communicating sequential processes, Communicaiton ACM, Vol.21, No.8, pp.666-676 (1978).

[24] Bertot, Y. and Casteran, P.: Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions, Springer (2004).

[25] Gordon, M. J. C.and Melham, T. F.: A Theorem Proving Environment for Higher Order Logic, Cambridge University Press (1993).

[26] Paulson, L. C.: Isabelle: A Generic Theorem Prover, Springer (1994).

[27] Holzmann, G. J.: The model checker SPIN, IEEE transaction on Software

Engineering, Vol.23, No.5 pp. 279-295 (1997).
[28] McMillan, K. L.: Symbolic Model Checking, Kluwer Academic Pub (1993).
[29] Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R. and Tacchella, A.: NuSMV Version 2: An Opensource Tool for Symbolic Model Checking, Proceeding of 14th International Conference on Computer Aided Verification, Vol.2404, Lecture Notes in Computer. Science, pp. 359-364 (2002).
[30] Larsen, G. K., Pettersson, P. and Yi, W.: UPPAAL in a nutshell, International Journal on Software Tools for Technology Transfer, Vol.1, No.1-2, pp.134-152 (1997).
[31] 中島震：ソフトウェア工学の道具としての形式手法 -彷徨える形式手法, NII Technical Report, NII-2007-007J（2007）.
[32] 中島震：オブジェクト指向デザインと形式手法，コンピュータソフトウェア，Vol.18，No.5，pp.499-528（2001）.
[33] 中島震：モデル検査法のソフトウェアデザイン検証への応用，コンピュータソフトウェア，Vol.23，No.2，pp.72-86（2006）.
[34] International Electrotechnical Commission (IEC): Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems, IEC 61508 (2010).
[35] International Organization for Standardization (ISO), Road vehicles -Functional safety, ISO 26262,(2011-2012).
[36] International Organization for Standardization (ISO)/ International Electrotechnical Commission (IEC): Information technology - Security techniques - Evaluation criteria for IT security, ISO/IEC 15408 (2008-2009).
[37] 独立行政法人情報処理推進機構：実務家のための形式手法厳密な仕様記述を志すための形式手法入門 第二版なぜ形式手法か, SEC-FM1-01-21 (2013).
[38] Clarke, E., Emerson, E. A. and Sistla, A. P.: Automatic verification of finite-state concurrent systems using temporal-logic specifications, ACM Trans. Programming Languages and Systems, Vol.8, No.2, pp.244–263 (1986).
[39] Peled, D.: Ten years of partial order reduction, Proceeding of Computer Aided Verification (CAV) 98, Lecture Notes in Computer Science, Vol.1427, pp.17-28 (1998).

[40] Alur, R. and Dill, D. L.: A theory of timed automata, Theoretical Computer Science, Vol.126, pp.183-235 (1994).

[41] Alur, R., Courcoubetis, C. and Dill, D. L.: Model checking in dense real time, Information and Computation, Vol.104, No.1, pp.2-34 (1993).

[42] Henzinger, T. A., Jhala, R., Majumdar, R. and Sutre, G.: Software Verification with BLAST, Proceedings of the Tenth International SPIN Workshop on Model Checking of Software, Vol.2648, pp.235–239 (2003).

[43] Henzinger, T. A., Jhala, R., Majumdar, R. and Sutre, G.: Lazy Abstraction, Proceedings of the 29th ACM Symposium on Principles of rogramming Languages, pp.58-70 (2002).

[44] Ball, T. and Rajamani, S.: The SLAM Toolkit, Proceedings of 13th Conference on Computer Aided Verification, Vol. 2102 of LNCS, pp.260-264 (2001).

[45] Havelund, K., and Pressburger, T.: Model checking JAVA programs using JAVA PathFinder, International Journal on Software Tool for Technology Transfer, Vol.2, No.4 (2001).

[46] Corbett, J. C., Dwyer, M. B., Hatcliff, J., Laubach, S., Pasareanu, C.S., Robby and Zheng, H.: Bandera: Extracting finite-state models from Java source code, Proceeding of 22nd International Conference on Software Engineering (2000).

[47] 篠崎孝一，太田弘，早水公二，星野光勇，今村哲典，吉田雅昭：モデル検査の実用化課題と支援ソフトウェアの開発，第三回システム検証の科学技術シンポジウム（2006）.

[48] 高田沙都子，森奈実子，村田由香里：モデル検査技術による上流工程での効率的な設計検証，東芝レビューVol.67, No.11, pp.45-49 (2012).

[49] 中島震：ソフトウェア工学からみたモデル検査法，第 22 回回路とシステム軽井沢シンポジウム（2009）.

[50] 中島震：SPIN モデル検査 -検証モデリング技法，近代科学社（2008）.

[51] 産業技術総合研究所システム検証研究センター:4 日で学ぶモデル検査（初級編），NTS（2006）.

[52] 吉岡信和，青木利晃，田原康之：SPIN による設計モデル検証 - モデル検査の実践ソフトウェア検証，近代科学社（2008）.

[53] Jacobson, J., Booch, I., Rumbaugh, G.: Unified Modeling Language Reference Manual，Addison-Wesley Professional（2004）.

[54] Latella, D., Majzik, I. and Massink, M.: Automatic verification of a behavioral subset of UML statechart diagrams using the SPIN model-checker, Formal Aspects of Computing, Vol.11, No.6, pp.637-664 (1999).

[55] Schafer, T., Knapp, A. and Merz, S.: Model Checking UML State Machines and Collaborations, Proceeding of Workshop on Software Model Checking, Vol.55, No.3 of Electronic Notes in Theoretical Computer Science (2001).

[56] Knapp, A., Merz, S. and Rauh, C.: Model checking timed UML state machines and collaborations, Proceeding of Seventh International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems, Vol.2469 of Lecture Notes in Computer Science, pp.395-416 (2002).

[57] Dwyer, B. M., Avrunin, S. G. and Corbett, C. J.: Pattern in Property Specification for Finite-State Verification, Proc. 21th International Conference on Software Engineering, pp.411-420(1999).

[58] Berezin, S., Campos, S., V., A. and Clarke, E. M.: Compositional reasoning in model checking, International Symposium on Compositionality 97 (COMPOS'97), Vol.1536 of Lecture Notes of Computer Science, pp.81–102 (1998).

[59] Peng, H. and Tahar, S.: Survey on compositional verification. Technical report, Department of Electrical and Computer Engineering, Concordia University (1998).

[60] Clarke, E. M., Long, D. E. and McMillan, K. L.: Compositional model checking, Proceedings of the Fourth Annual Symposium on Logic in Computer Science, pp.353-362 (1989).

[61] Grumberg, O. and Long, D.: Model checking and modular verification, ACM Transactions on Programming Languages and Systems, Vol.16, No.3, pp.843-871 (1994).

[62] Cobleigh, J. M., Giannakopoulou, D. and Pasareanu. C.S.: Learning assumptions for compositional verification, In Proc. 9th TACAS, LNCS 2619, Springer, pp.331–346 (2003).

[63] Henzinger, T. A., Qadeer, S. and Rajamani, S. K.: You assume, we guarantee: methodology and case studies, In CAV 98: Computer Aided Verification, LNCS 1427, Springer-Verlag, pp.440-451 (1998).

[64] Boehm, W. B.: Software Engineering Economics, pp.38-40, PRENTICE-HALL,

USA (1981).

[65] Betriebliches Lastenheft fur Funkfahrbetrieb. stand 1.10 (1996).

[66] Hansel, F., Poliak, J., Slovak, R. and Schnieder, E.: Reference Case Study ”Traffic Control Systems” for Comparison and Validation of Formal Specifications Using a Railway Model Demonstrator, Integration of Software Specification Techniques for Applications in Engineering, pp.96–118 (2004).

[67] Klop, J. W.: Term rewriting systems, Handbook of Logic in Computer Science, Volume 2. Background: Computational Structures, Oxford University Press, Oxford, UK, pp.1–116 (1992).

[68] Henry, S. and Faure, M. J.: Elaboration of invariants safety properties from fault-tree analysis, In Proceeding of IMACS-IEEE Computational Engineering in Systems Applications (2003).

[69] Barragan, S. I. and Faure, M. J.: From Fault Tree Analysis to Model Checking of logic Controllers, In Proceeding of 16th International Federation of Automatic Control World Congress (2005).

[70] The National Aeronautics and Space Administration: Fault Tree Handbook with Aerospace Applications, no number version 1.1(2002).

[71] U.S. Nuclear Regulatory Commission: Fault Tree Handbook, NUREG-0492 (1981).

[72] 小野寺勝重：国際標準化時代の実践 FTA 手法－信頼性，保全性，安全性解析と品質保証，pp.115-131，日科技連（2000）.

[73] Yan, B., Nakamura, M. and Matsumoto, K.: Deriving Safety Properties for Home Network System Based on Goal-Oriented Hazard Analysis Model, International Journal of Smart Home, Vol. 3, No. 1, pp.67–79 (2009).

[74] Dardenne, A., Lamsweerde, V., A. and Fickas, S.: Goal-directed Requirements Acquisition, Science of Computer Programming, Vol.20, pp.3-50 (1993).

[75] Lamsweerde, V., A.: Goal-Oriented Requirements Engineering: A Guided Tour, Proceeding of 5th IEEE International Symposium on Requirements Engineering, pp.249-263 (2001).

[76] Kletz, T., A.: HAZOP and HAZAN: Identifying and Assessing Process Industry Hazards, The Institution of Chemical Engineers (2006).

[77] 前島弘則，神武直彦：Dual Vee Model を用いた根本原因解析手法

（RCADVM）の構築と適用，慶應義塾大学大学院システムデザイン・マネジメント研究科博士論文（2013）.

[78]Forsberg, K., Mooz, H. and Cotterman, H.: Visualizing Project Management, John Wiley & Sons, Inc. (2005).

[79]日本規格協会：対訳 ISO9001 品質マネジメントの国際規格，日本規格協会（2009）.

[80]清木昌：ゲームシナリオのモデル検査，情報処理学会研究報告（ゲーム情報学），Vol.2004，No.28，pp.51-56（2004）.

研究業績

1. 定期刊行誌掲載論文（主論文に関連する原著論文）
   (1) Atsushi Katoh, Shinichiro Haruyama, Naohiko Kohtake and Yoshiaki Ohkami: Model Checking of Safety Properties for Complex Systems Using MWF Inactivation and MNWF Activation, International Journal of Computer Science and Mobile Computing, Vol.2, No.12, pp.26-39 (2013).
   ※ Best paper of the month Award 受賞
   (2) 加藤淳，浦郷正隆，狼嘉彰：構成要素の連携動作に着目した産業用ロボットの開発：IEEE 1220 におけるアーキテクチャ設計プロセスおよび UPPAAL によるモデル検査を融合した開発手法の提案, 計測自動制御学会産業論文集，Vol.10，No.5，pp.37-49（2011 年）.
   (3) 加藤淳，浦郷正隆，狼嘉彰：複雑システムの信頼性を向上させる開発手法：アーキテクチャ設計手法とモデル検査の融合，Synthesiology，Vol.3，No.3，pp.197-212（2010 年）.

2. 国内学会発表
   (1) 加藤淳*，松本充広，春山真一郎：MWF/MNWF に基づく FTA と Spec Patterns によるモデル検査式の導出，情報処理学会論文誌プログラミング（PRO），Vol.5，No.4，pp.40（2012 年）.
   (2) 加藤淳*，神武直彦，春山真一郎，狼嘉彰：モデル検査を用いて組込みシステムにおけるソフトウェアとハードウェアの協調動作に関する要求仕様の不整合を検出する手法，組込みシステムシンポジウム 2009，IPSJ Symposium Series Vol. 2009，pp.65-70（2009 年）.
   (3) 加藤淳*，狼嘉彰 : FPGA とソフトウェアにおける協調動作の整合性に関する評価手法の提案，情報処理学会 第 163 回ソフトウェア工学研究会研究報告，Vol.2009，No.31，pp.105-112（2009 年）. **※研究賞受賞**

謝辞

本博士学位論文は，筆者が慶應義塾大学大学院システムデザイン・マネジメント（SDM）研究科後期博士課程在学中に，ユビキタス通信ラボおよびストラテジック･デザインラボで行った研究をまとめたものです.

ユビキタス通信ラボに在籍の間, 本研究に関して熱心にご指導ご鞭撻を頂きました，本論文の主査である本学春山真一郎教授に心より感謝いたします．教授からは，特に研究の成果を明確にアピールすることの重要性をご指導頂きました．また，本研究を俯瞰する多くのコメントを頂きました．教授との打ち合わせの度に，本研究が更に進化することを実感することができました．ありがとうございました.

ストラテジック･デザインラボに在籍の間，本研究に関して熱心にご指導ご鞭撻を頂きました, 本論文の副査である狼嘉彰前 SDM 研究科委員長に心より感謝いたします．前研究科委員長と初めてお会いしたのは，筆者が在籍する有人宇宙システム株式会社において，前研究科委員長が技監に就任されていたときに遡ります．前研究科委員長の人間性，ご教示頂いたシステムズエンジニアリングの魅力が無ければ, 本論文は無かったと思います. ありがとうございました.

本論文をご精読頂き有用なコメントを頂きました, 本論文の副査である本学神武直彦准教授に感謝いたします．准教授は，准教授が宇宙航空研究開発機構に在籍されているとき，本学への入学を検討している筆者の背中を押してくださいました．また，学位のすばらしさを，身を以って示してくれた方です．ありがとうございました.

本論文をご精読頂き有用なコメントを頂きました, 本論文の副査である筑波大学大学院システム情報系情報工学域の追川修一准教授に感謝いたします．准教授には，筆者が本学に入学する以前よりお世話になっていました．本学への入学後も，准教授が運営するシンポジウムにお誘い頂き，研究を発表する機会を頂きました．また，本論文の審査においては，多くの鋭いご指摘を頂きました．ご指摘のお答えに詰ってしまうことが多々ありました．おかげさまで，よい博士学位論文に仕上がりました．ありがとうございました.

本論文の 3 章の内容については，当時，ストラテジック･デザインラボに在籍されていた浦郷正隆氏から多くの貴重なコメントを頂きました．心より感謝いたします．本論文の 4 章の内容については，有人宇宙システム株式会社の筒井

良夫氏，松本充広氏，川口真司氏から多くの貴重なコメントを頂きました．心より感謝いたします．また，ストラテジック･デザインラボの杢野正明氏，安岡寛道氏をはじめ，多くの本学の学生各位から，社会人学生としての心得，研究の進め方などの貴重なアドバイスを頂きました．心より感謝いたします．

本論文の執筆や審査にあたり，筆者の休暇を快く認めて頂き，また，筆者の研究活動を暖かく見守ってくださった，有人宇宙システム株式会社および三菱航空機株式会社の皆様に心より感謝いたします．今後，本研究の成果を業務に生かすことで恩返しさせて頂きます．ありがとうございました．

最後に，本論文の完成まで筆者を支えてくれた，妻のあづさ，長女の諒，次女の英，長男の鋼に，この場を借りて深謝いたします．ありがとうございました．

平成 26 年 3 月

加藤　淳