社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

反例を利用した網羅性の高いプロパティ集合生成手法

松本　剛史† 　李　蓮福†† 　吉田　浩章† 　余宮　尚志††† 　藤田　昌宏†

† 東京大学大規模集積システム設計教育研究センター　〒 113–0032 東京都文京区弥生 2–11–16
†† 東京大学大学院工学系研究科電気系工学専攻　〒 113–8656 東京都文京区本郷 7–3–1
††† 株式会社東芝 ソフトウェア技術センター　〒 212–8582 神奈川県川崎市幸区小向東芝町 1
E-mail: matsumoto@cad.t.u-tokyo.ac.jp, lee@cad.t.u-tokyo.ac.jp, hiroaki@cad.t.u-tokyo.ac.jp,
hisashi.yomiya@toshiba.co.jp, fujita@ee.t.u-tokyo.ac.jp

あらまし　ソフトウェアとハードウェアが協調動作する大規模なシステムの仕様設計段階における検証は、以降の設計において誤りが生じることを防ぐために非常に重要であるが、仕様記述の不明確さなどのために、検証するプロパティを常に正しく記述することは難しい。本研究では、検証を考慮したシステムの仕様設計段階における設計フローを示すとともに、その設計や後工程の設計を検証するためのプロパティ集合の導出について述べる。一般的に、プロパティを用いた検証では、プロパティの網羅性によって検証の質が決定されるため、プロパティの網羅性の指標、および、より網羅的なプロパティを生成するための工夫・手法が不可欠である。本研究では、有限状態機械として記述された仕様に対して、プロパティの検証時にたどられる状態遷移の割合による網羅性の指標を提案する。また、提案する設計フローにおいて、シミュレーション等を通して、正しくないと判断される実行例からプロパティを生成する手法をエレベータ制御の例題を通して示す。

キーワード　仕様、上位設計、形式的検証、プロパティ検証、UML

Generation of High Coverage Property Set Using Counterexamples

Takeshi MATSUMOTO†, Yeonbok LEE††, Hiroaki YOSHIDA†, Hisashi YOMIYA†††, and
Masahiro FUJITA†

† VLSI Design and Education Center, University of Tokyo
2–11–16 Yayoi, Bunkyo-ku, Tokyo, 113–0032 Japan
†† Dept. of Electrical Engineering and Information Systems, University of Tokyo
7–3–1 Hongo, Bunkyo-ku, Tokyo, 113–8656 Japan
††† Corporate Software Engineering Center, Toshiba Corporation
Komukai-Toshiba-cho 1, Saiwai-ku, Kawasaki, Kanagawa, 212–8582 Japan
E-mail: matsumoto@cad.t.u-tokyo.ac.jp, lee@cad.t.u-tokyo.ac.jp, hiroaki@cad.t.u-tokyo.ac.jp,
hisashi.yomiya@toshiba.co.jp, fujita@ee.t.u-tokyo.ac.jp

Abstract　When we design a large system including hardware and software, verification in specification-level is very important to avoid design bugs from the subsequent design-levels such as system-level and behavior-level. However, partly because the specification documents itself has unclarity and ambiguity, correctly writing properties to be kept in designs is difficult. In this work, we propose a high-level design flow considering verification and property derivation in that flow. In general, the quality of property checking is decided by the completeness of a set of given properties. Therefore, metrics to measure the completeness of the property set and methods to generate properties with high completeness are required. We propose a transition-based coverage metric that is defined in finite state machine of the system specification. Through a case study of an elevator controller design, we also show a way to generate properties that can improve the coverage from counterexamples found in simulation.

Key words　Specification, High-level design, Formal verification, Property checking, UML

1. はじめに

多くのソフトウェアモジュールとハードウェアモジュールから成る大規模なシステムを効率的に設計するために、UML(Unified Modeling Language: 統一モデリング言語) [1] などを用いた仕様レベルからの設計が行われている。そのような設計においては、与えられた仕様記述からシステム全体の動作や構造を高い抽象度で設計し、その後、各モジュールに対してより詳細な設計を進めていく。文献 [2], [3] では、ソフトウェア・ハードウェア協調システムの UML を用いた上位設計方法論が提案されている。

プロパティ検証とは、設計が満たすべき性質 (プロパティ) を実際に設計が満たしているかどうかを検証する手法であり、ソフトウェア設計やハードウェア設計の抽象度の高い段階において広く用いられている。プロパティ検証を行うためには、数学的な表現 (多くの場合、時相論理式) によって検証する性質を記述することが必要である。そのため、多くの場合において不明確さや曖昧さを持つ自然言語等で記述された仕様記述から仕様設計を行う段階においては、プロパティの記述自体が不可能な性質がある。また、プロパティ検証においては、プロパティとして記述されていない性質に関しては検証することができない。そのため、より網羅性の高いプロパティの集合を作ることが必要である。

自然言語として記述された仕様記述から設計仕様を決定する段階の設計においては、検証を考慮することも重要となる。これは、抽象度の高い段階での設計は、それ以降の設計に対してゴールデンモデルとして利用されるため、設計誤りがある場合に設計やり直しや誤り修正のために多大なコストがかかる可能性が高いためである。そこで、本研究では、検証の適用を考慮した仕様設計フローを提案する。この設計フローでは、通常のユースケース図・シーケンス図・クラス図といった UML 図を用いた解析を通して、システムの動作を有限状態機械として表す。この有限状態機械は、後工程の設計に対するゴールデンモデルとして参照することが可能である。また、設計が満たすべきプロパティはこの有限状態機械上で記述される。そのため、十分なプロパティの集合がこの有限状態機械上で記述されていれば、それを後工程の設計に対して適用し、検証を行うことができる。

また、前述のとおり、プロパティ検証によって設計の正しさを検証する場合、プロパティの網羅性が高いことが求められる。しかし、一方で、与えられたプロパティ集合の網羅性を評価するための指標や網羅性の高いプロパティを生成する手法に関する研究はあまりなされていない。本研究では、有限状態機械に対して検証されたプロパティがたどった遷移が、全体の遷移を網羅している割合によって、プロパティ集合の網羅性を測定する指標を提案する。

本論文の構成は以下の通りである。第 2 節では、UML 等を用いる設計の初期段階における検証を扱う関連研究を紹介する。第 3 節では、提案する検証を考慮した設計フローを述べ、加えて、与えられたプロパティ集合の網羅性を測定する指標、検証で得られた反例から網羅性を高めるプロパティを生成する手法を述べる。第 4 節では、エレベータ制御システムを例としたケーススタディを紹介する。第 5 節でまとめと今後の課題を示す。

2. 関連研究

本節では、UML 等を用いたシステム設計の初期段階における検証に関する研究を紹介する。

2.1 テストパタン生成

文献 [4] において、UML によって記述された設計から自動的に検証シナリオ (事前条件、事後条件、不変条件) を抽出する手法が提案されている。また、文献 [5] では、階層的なシーケンス図を用いて検証に利用することができるテストパタンを自動的に生成する手法を提案している。この手法では、全てのシーケンス図を一度は含むようなテストパタンの集合が自動的に生成することができる。また、文献 [6] では、ステートチャート図を利用した並行プログラムのテストシーケンス生成手法を提案している。この手法では、ステートチャート図上でイベント発生とそれらの間の依存関係を特定し、多数存在する可能なスケジューリングから 1 つを選び、テストシーケンスとすることによって、並行プログラムの検証を可能としている。

2.2 完全なプロパティ集合の生成

文献 [7] では、検証する 1 つのシーケンスを複数のトランザクションに分割し、そのそれぞれについてプロパティを記述する手法を提案している。このとき、前のプロパティの入出力変数や主要な状態変数の事後条件と後のプロパティの事前条件が矛盾しないような一連のプロパティを初期状態から記述することによって、検証するシーケンスについて完全な検証を行うことができる (図 1)。

2.3 設計の無矛盾性チェック

仕様設計段階では、設計は 1 つの設計言語で記述される場合は少なく、システムを複数の異なる観点から見て設計・解析を進めることが多い。広く行われている UML を用いる設計においても、複数の UML 図を用いて設計が進められることが通常である。そのため、異なる観点間の無矛盾性のチェックが必要である。文献 [8] では、シーケンス図と状態遷移図を用いて記述されたシステムがシーケンス図に記述された通りの動作を行うかどうかをモデル検査を用いて検証する手法を提案している。また、文献 [9] では、XML(eXtensible Markup Language) によって記述された仕様内で矛盾がないかどうかを自動チェックする手法を提案している。

以上で述べた 3 つのアプローチのうち、本研究では 2 番目で述べた文献 [7] のように、プロパティ集合を生成することによって設計検証を行うことを目指している。RTL 設計からプロパティを生成して検証を行う文献 [7] の手法とは異なり、設計の初期段階で可能な限り完全性の高いプロパティの集合を作成し、それを後工程の設計の検証に利用することも目的としている。

3. 提案するプロパティ生成と検証

本節では、仕様設計段階でプロパティ検証を行うための検証

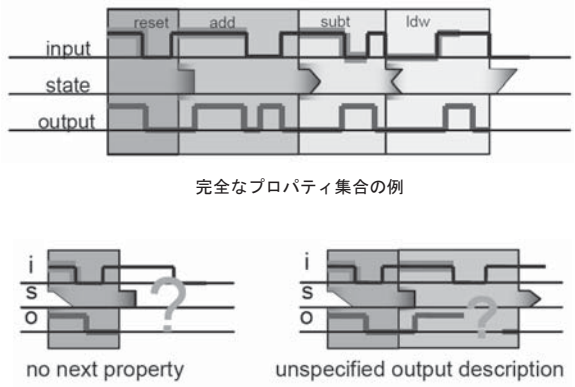


図 1　完全なプロパティ集合の概念 [7]

フロー、プロパティの網羅性指標、反例を利用したプロパティ生成について述べる。

3.1　設計フロー

本研究で提案する検証を考慮した上位設計フローを図 2 に示す。提案する設計フローでは、自然言語で記述された仕様記述から設計を始め、ユースケース分析、振る舞い詳細化を行うことによって振舞い記述を得る。この段階では、システムの振舞い (動作) のみが決定されており、構造については全く何も決まっていない。その後、階層化や Processing Element への割り付けを行い、システムレベル設計記述を得る。本設計フローの出力となるシステムレベル設計記述は、SystemC 言語のような C ベース設計言語で記述されることを想定している。

本研究では、この設計過程におけるシステムの動作を決定する振舞い詳細化の際に、振舞いモデルと呼ばれる振舞いの仕様を有限状態機械 (FSM: Finite State Machine) で記述することを取り入れている。この振舞いモデルは、後工程の設計記述に対して、ゴールデンモデルとして参照することができるものである。そのため、後工程の設計記述の検証は、

- 振舞いモデルとの間の等価性検証
- 振舞いモデルで成立するプロパティ集合を用いたプロパティ検証

によって実現することができる。本研究では、このうち、プロパティ検証を行うことを目的としている。そのためには、振舞いモデルが満たすべきプロパティを記述する必要がある。また、上述のプロパティ生成と同時に、振舞いモデルをゴールデンモデルとして検証に用いるためには、振舞いモデル自体の検証も非常に重要である。

図 3 は、仕様記述から振舞いモデルを生成する過程を示したものである。ここで行われる作業は、通常の UML 等を用いた仕様設計で行われるものとほぼ同じである。ただし、最終的な出力は、システムのアスペクトごとに記述された並列動作する FSM となる。そのため、システム全体の FSM は、それらの FSM の積として表現される。

3.2　プロパティの網羅性

本節では、与えられたプロパティが検証対象の FSM をどの程度網羅しているかを評価する指標を提案する。

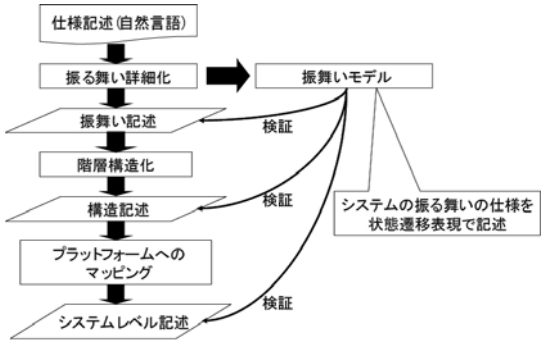


図 2　検証を考慮した上位設計フロー

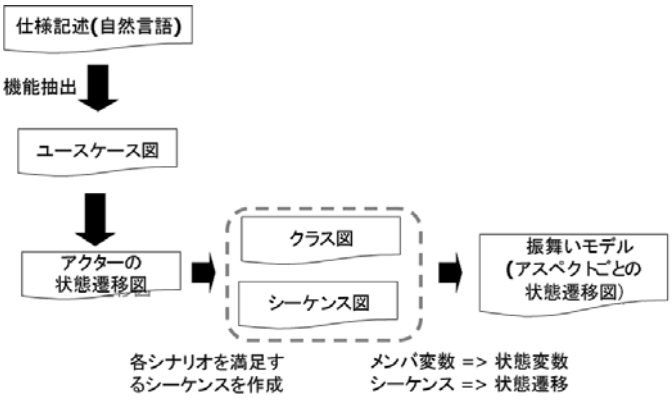


図 3　振舞いモデル生成過程

提案する指標は、検証に成功したプロパティが成立するために通る FSM 上の全ての遷移列に含まれている状態遷移の、FSM に含まれる全ての状態遷移に対する割合として定義する。

本研究では、P を与えられたプロパティの集合とするときプロパティ $P_i \in P$ は、以下の形で書けることを仮定する。

$$P_i : \{S_S\}\&\&\{I_{seq}\} \rightarrow X^N(\{S_E\})\&\&\{O_{seq}\} \qquad (1)$$

ここで、S_S はプロパティの前提条件となっている状態の集合、S_E はプロパティ成立後の事後条件となっている状態の集合、I_{seq}, O_{seq} は、それぞれプロパティが成立するための入力条件とプロパティ成立時の出力条件である。また、X^N は N サイクル後を表す。

検証の対象となっている FSM は、$(I, O, S, s_0, \delta, \lambda)$ で定義される。ここで、I, O は入力と出力、S は有限状態集合、s_0 は初期状態の集合、δ, λ は状態遷移関数と出力関数である。このとき、この FSM の状態遷移関係 $((S, S) \rightarrow \{true, false\})$ が含む最小項の数が、FSM に含まれる遷移の総数となる。これを TR_{fsm} とする。

また、一つのプロパティ P_i が成り立つ際に、S_S から S_E に至る遷移列が複数存在する。その遷移列に含まれる遷移の総数を TR_{prop} とする。このとき、プロパティ P_i の網羅性は TR_{prop}/TR_{fsm} であると定義する。プロパティ集合 P の網羅性は、それぞれのプロパティにおいて得られた遷移のうち重複しないものに対して定義されるものとする。

ここで、例として、図 4 に並列に動作する 2 つの FSM において、与えられたプロパティの網羅性を求める過程を示す。な

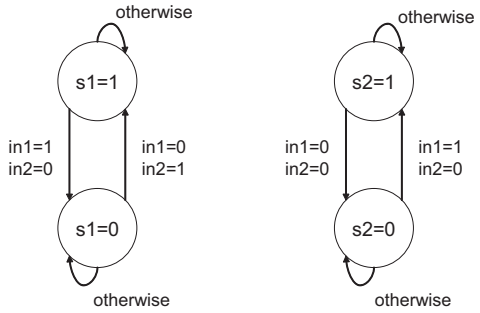


図 4　例

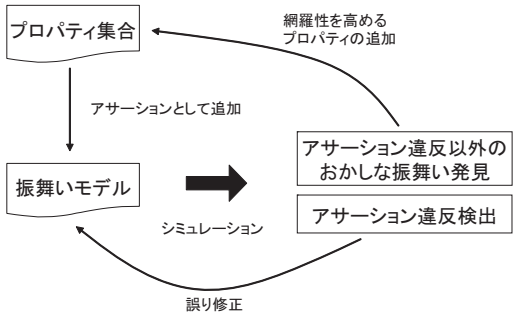


図 5　反例を利用したプロパティ追加方法

お、2 つの FSM は同期して動作しているとする。今、プロパティ$P1 : s1 = 1\&\&in2 = 0 \rightarrow X(s1 = 1)$ の検証を考える。このプロパティは FSM 上で成立する。このとき、$P1$ が通る遷移は、

（1）$(s1 = 1, s2 = 1) \rightarrow (s1 = 1, s2 = 0)$

（2）$(s1 = 1, s2 = 0) \rightarrow (s1 = 1, s2 = 0)$

の 2 つである。検証している FSM の全状態遷移数は 16 であるため、$P1$ の網羅性は 2/16 となる。次に、プロパティ$P2 : s1 = 0\&\&(in1 = 1\&\&X1(in1 = 1)) \rightarrow X2(s1 = 0)$ を検証する。このプロパティは FSM 上で成立する。ここで、$P2$ が通る遷移を列挙すると、

（1）$(s1 = 0, s2 = 0) \rightarrow (s1 = 0, s2 = 0)$
$\rightarrow (s1 = 0, s2 = 0)$

（2）$(s1 = 0, s2 = 0) \rightarrow (s1 = 0, s2 = 0)$
$\rightarrow (s1 = 0, s2 = 1)$

（3）$(s1 = 0, s2 = 0) \rightarrow (s1 = 0, s2 = 1)$
$\rightarrow (s1 = 0, s2 = 1)$

（4）$(s1 = 0, s2 = 1) \rightarrow (s1 = 0, s2 = 1)$
$\rightarrow (s1 = 0, s2 = 1)$

となる。この中に含まれている遷移は 3 種類であるため、網羅性は 3/16 となる。

ここで定義された網羅性指標は、ある状態変数 S_i について、その変数が常に同じ値になるようなプロパティを加えているだけでは向上しないことになる。そのため、この網羅性指標を高めるためには、検証対象の FSM に含まれる状態変数が特定の値に偏ることのないようなプロパティを記述する、または、追加する必要がある。

3.3　反例を利用したプロパティ生成

従来のプロパティ検証では、検証するプロパティは設計者や検証者が仕様記述等を参照しながら考えて記述されてきた。そのため、仕様記述に不明確さや曖昧さがある場合には、その性質をプロパティとして記述することが困難となっている。また、仕様記述から生成可能なプロパティを全て列挙することは人手では難しい。加えて、仕様記述に漏れがある可能性もある。

このような状況でプロパティを生成する手段の一つとして、シミュレーション中に発見されたおかしな振舞いから、その振舞いが明確に反例と判断できるようなプロパティを追加することを行う。その手順を図 5 に示す。まず、仕様記述等を参照することによって既に得られているプロパティをアサーションとして設計記述に組込む。このアサーションに違反する振舞いが発見された場合、元のプロパティが正しい限り、振舞いモデルに誤りがあるため、それを修正する。いずれのアサーションにも違反していないが、おかしな振舞い (誤りである可能性が高い振舞い) が発見された場合、その振舞いを誤りとするかどうかを判断する。誤りと判断された場合、その振舞いを誤りと判断するためのプロパティが不足していたこととなる。そこで、その誤った振舞いに対するプロパティを作成する。このとき、複数の候補がある場合には、前節で述べた網羅性が高くなるようなプロパティを選択する。この作業を繰り返すことによって、網羅性のより高いプロパティ集合を生成していくことができる。

4.　ケーススタディ

本節では、これまでに述べた手法を簡単なエレベータ制御システムに対して行ったケーススタディとその結果を示す。

4.1　エレベータ制御システム

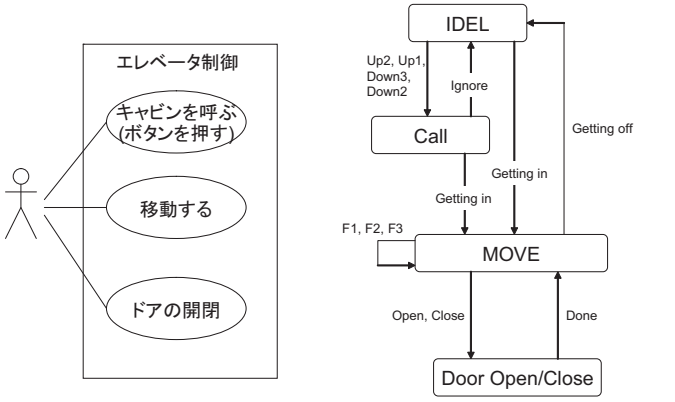
例題として用いたエレベータ制御システムは以下の性質・特徴を持つ。

- 3 階建ての建物で動作するエレベータ 1 基の制御を行う
- キャビン内に階数ボタン (F1, F2, F3) があり、押されるとその階へ向かう
- 各フロアには上下ボタン (1 階は上ボタン、3 階は下ボタンのみがある) があり、押されるとキャビンが要求された方向へ向かう場合、その階で停止する
- 要求された階で停止した場合、エレベータのドアが自動的に開く
- 閉ボタンを押すとキャビンのドアが閉まる
- キャビンはドアが閉まった状態でのみ、要求があった階へ向かって動くことができる

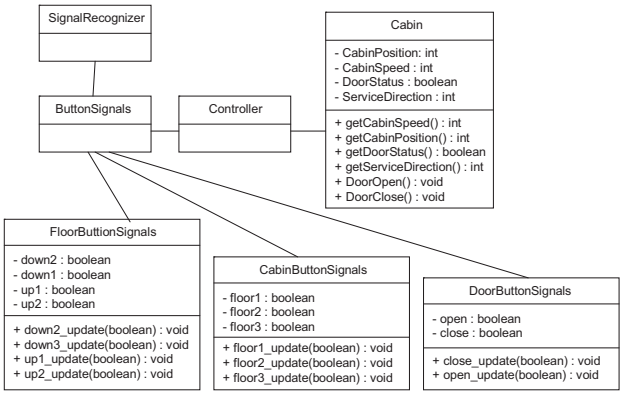
以上の性質・特徴は全て仕様記述に記載されている。仕様記述には、上述のような性質・特徴の他に、システムに対する入出力、これ以外の基本的な動作のシナリオも仕様記述として記載されている。本ケーススタディでは、この仕様記述を出発点として仕様設計、振舞いモデル作成、検証を行った。

4.2　振舞いモデルの作成

仕様記述をもとに、振舞いモデルの作成を行った。図 6 に、その過程で記述された UML 図を示す。ユースケース図、シーケンス図、クラス図を用いて通常のシステム設計と同様の解析を行いながら、仕様記述を実現するような振舞いを作成してい



(a) ユースケース図　(b) アクタの状態遷移図



(c) クラス図

図 6　設計過程で記述された UML 図の例

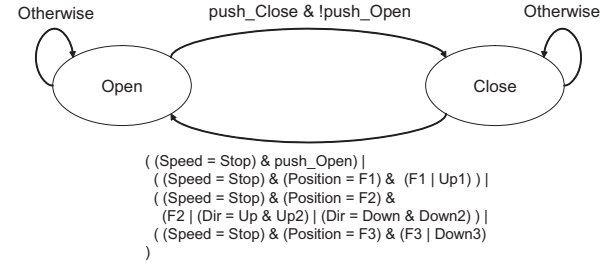
く。加えて、本研究では、第 3.1 節で述べたように、アクターの状態遷移図を作成する。これは、あるアクターが取り得る状態とその遷移を図示したものであり、各ユースケース間の関係や網羅的なテストパタン作成に利用することができる。

本ケーススタディでは、クラス図を参照しながら、ドア開閉・キャビン動作・キャビン位置・キャビン運行方向・フロアボタン・キャビン内の階数ボタンの計 6 つのアスペクトを選び、それぞれについて FSM を作成した。その例を図 7 に示す。図では、ドア開閉およびキャビン内の階数ボタンについて、その振舞いを FSM で記述している。今回のケーススタディでは、並列動作する各アスペクトの FSM の状態遷移は同期して起こるとした。

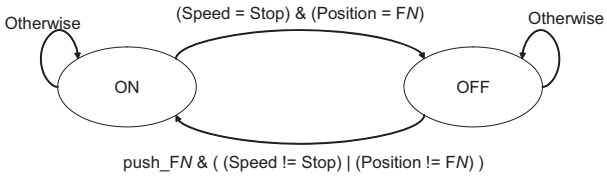
得られたエレベータ制御システムの振舞いモデルは、以下のようになった。

- 11 個の FSM が並列動作 (階数ボタンとフロアボタンのアスペクトに属するインスタンスが複数あるため)
- 全体で 11520 状態の FSM
- 状態遷移数
  - 実行可能性を考慮しない場合 3.5×10^7
  - 実行可能なもののみ 6.5×10^3

なお、11520 状態の FSM において理論上あり得る遷移の数は、$(11520)^2 = 1.3 \times 10^8$ である。実行可能性を考慮しない場合については、各 FSM にある遷移の積を取ったものである。また、実行可能な遷移の数は、SIS [10] を用いて真になり得る状態遷



(a) ドア



(b) キャビン内の N 階ボタン

図 7　振舞いモデル

移関係の最小項数を数えたものである。

4.3 網羅性の測定

まず、エレベータ制御システムにおいて成立するべきいくつかのプロパティについて、その網羅性を測定した。

a) プロパティ 1

$(Door = Open)\&\&push\_Close\&\&!push\_Open$
$\rightarrow X(Door = Close)$

このプロパティは、キャビンのドアが開いた状態から閉まるための条件を記述したものである。例題のエレベータでは、開ボタンの方が閉ボタンよりも優先するため、ドアが閉まる際には開ボタンが押されていないことが条件となっている。このプロパティは設計で満たされていることが確認された。このプロパティが満たされるときに、前提条件となっている状態 $(Door = Open)$ から事後条件となっている状態 $(Door = Close)$ に至る遷移は、768 個あった。そのため、このプロパティが網羅している遷移の割合は 12%となる。

b) プロパティ 2

$(Position = F1\_F2)\&\&(Direction = Up)\&\&U2$
$\rightarrow X(Speed = Stop)$

このプロパティは、キャビンが 1 階と 2 階の間にあって上向きに運行している場合、2 階のフロアで上向きボタンが押されている $(U2)$ とキャビンが停止することを示している。この設計では、時間や距離などの物理量は抽象化してモデル化されており、1 サイクルで階と階の間から次の階へ到達することができるようになっている。このプロパティについても同様に遷移数を調べたところ、121 個であった。そのため、このプロパティの網羅性は 1.8%になる。このように、検証するプロパティによって、FSM 上での正しい動作に含まれる遷移数は大きく異なっている。

また、今回のケーススタディにおいて、振舞いモデルを SpecC 言語 [11] に変換してシミュレーションしたところ、それまでに記述されたプロパティには違反していないが、意図していない動作が観測された。その動作は「初期状態 (キャビンのドアが

閉まっており、1 階で停止していてどのボタンも押されていない状態) から始めて、3 階の下向きボタンを押したところ、キャビンが上昇しなかった」というものであり、設計誤りであった。そこで以下の 3 つのプロパティの追加を検討した。

- キャビンが 1 階に停止していて 1 サイクル後にドアが閉まっているとき、3 階の下向きボタンが押されると、1 サイクル後にキャビンが上昇する
- キャビンが 1 階に停止していて 1 サイクル後にドアが閉まっているとき、2 階以上のボタンが 1 つ以上押されていると 1 サイクル後に上昇する
- キャビンが 1 階に停止していて 2 階以上のボタンが 1 つ以上押されていれば、1 サイクル後にキャビン運行方向が上向きとなる。かつ、キャビン運行方向が上向きのとき、その 1 サイクル後にドアが閉まっていれば、キャビンが上昇する

このうち、最も網羅性が高いものは 3 番目のプロパティであるため、これを追加することとした。加えて、このプロパティに相当する仕様を仕様記述に追加した。

5. まとめと今後の課題

本稿では、システムの仕様設計段階における検証を考慮した設計フローの提案とそのフロー中におけるプロパティ検証の適用について議論した。プロパティ検証においては、いかに網羅性に優れたプロパティの集合を用意できるかによって、検証の品質が大きく影響される。そのため、本稿では、プロパティの網羅性の指標として、プロパティが満たされているときに、有限状態機械として表現された設計が通る可能性のある遷移の割合を用いることを提案した。予備的な結果として、エレベータ制御システムに対するいくつかのプロパティについて、網羅性を測定した。また、シミュレーションによって得られた意図しない動作のトレースから、それが反例となるようなプロパティを追加する手法も提案した。

今後の課題としては、以下の項目が挙げられる。

- 本稿では、提案した網羅性指標がどの程度、実際の設計誤り検出に有効であるかの評価を行うことができなかった。今後、より実用的な例題に対して、網羅性指標と設計誤り検出の評価を行っていく。
- 本研究では、動作のゴールデンモデルである振舞いモデルから作成したプロパティを後工程の検証に用いることを最終的な目標としている。そのため、得られたプロパティによって下位設計が検証できるように、同じ性質を抽象度の低い記述で記述し直す必要がある。
- 本研究では、初期段階の設計を対象としているため、設計の抽象度が高く、扱う FSM の状態数はそれほど多くないことを想定している。しかし、後工程の設計に提案した網羅性指標を適用するためには、より効率的に算出が可能な網羅性指標を検討する必要がある。

文　献

[1] Object Management Group – UML http://www.uml.org/

[2] K. Matsui and M. Fujita, “Object-Oriented analysis and specification for HW/SW co-design with UML diagrams,” *Prof. of IASTED International Conference on Advances in Computer Science and Technology*, pp.38–43, Jan. 2006.

[3] 李蓮福, 石川悠司, 小島慶久, 吉田浩章, 余宮尚志, 小松聡, 藤田昌宏, “既存設計の再利用を考慮した SoC の仕様記述手法と上位設計方法論,” 電子情報通信学会技術研究報告, Vol.107, No.505, pp.49–54, 2008 年 3 月.

[4] 大石亮介, 松田明男, 岩下洋哲, 高山浩一郎, “仕様書から検証シナリオを生成する手法,” 情報処理学会研究報告, 2006-SLDM-127, pp.1–4.

[5] P. Murthy, S. Rajan, and K. Takayama, “High Level Hardware Validation Using Hierarchical Message Sequence Charts,” *Proc. of Ninth IEEE International High-Level Design Validation and Test Workshop*, pp.167–172, Nov. 2004.

[6] H. Seo, S. Chung, and Y. Kwon, “Generating Test Sequences from Statecharts for Concurrent Program Testing,” *IEICE Trans. on Inf. and Syst.*, Vol.E89-D, No.4, pp.1459–1469, Apr. 2006.

[7] J. Bormann, S. Beyer, A. Maggiore, T. Blackmore, and F. Bruno, “Complete Formal Verification of TriCore2 and Other Processors,” *Proc. of Design and Verification Conference 2007*, Feb. 2007.

[8] 宮崎仁, 横川知教, 佐藤貞仁, 佐藤洋一郎, 早瀬道芳, “有界モデル検査を用いた複数 UML 図の形式的検証,” 電子情報通信学会技術研究報告, Vol.107, No.505, pp.13–18, 2008 年 3 月.

[9] 石川悠司, S. Kang, 李蓮福, G. Park, 渡邊翔太, 瀬戸謙修, 小松聡, 浜村博歴史, 藤田昌宏, “ハードウェア設計における設計資産の仕様記述およびその検証手法,” 電子情報通信学会技術研究報告, Vol.106, No.547, pp.43–48, 2007 年 3 月.

[10] E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, R.K. Brayton, and A. Sangiovannni-Vincentelli, “SIS: A system for sequential circuit synthesis,” 1992.

[11] D. Gajski, J. Zhu, R. Doemer, A. Gerstlauer, and S. Zhao, *SpecC: Specification Language and Methodology*, Kluwer Academic Publisher, March 2000.