ミッションクリティカルシステムの形式検証技術

Formal Verification Techniques for Mission-Critical Systems

大須賀　昭彦　A. Ohsuga
林　俊文　T. Hayashi
神山　雅彦　M. Kamiyama

交通，金融，医療，プラント制御などのシステムのように，機能の確実な実行が特に重要視されるシステムをミッションクリティカルシステムと呼ぶ。近年，ミッションクリティカルシステムの信頼性を確保する技術として形式検証技術（Formal Verification Techniques）が注目されている。形式検証とは，システムが仕様どおりに正しく動作することを数学的な方法によって証明する技術であり，今後ますます大規模・複雑化すると考えられるこの種のシステムに対し，高い精度での信頼性が保証できるものとして期待されている。この形式検証技術について当社は，原子力プラント制御やその他の分野への適用を試み，技術の有効性を確認するとともに実用化の見通しを得た。

Systems whose reliability is regarded as most important, such as traffic, financial, medical, and plant control systems, are called mission-critical systems. Formal methods, especially formal verification, are being investigated as promising ways to increase the reliability of these systems. Formal verification uses mathematical techniques to assure the correctness of a system; that is, to prove that a system meets its specifications.

This paper describes the use of formal verification techniques, and outlines some examples of the application of formal verification to mission-critical systems.

1 まえがき

交通，金融，医療，プラント制御などのシステムのように，誤動作によって何らかの被害や損害を生む恐れをもち，目的の任務（ミッション）を果たすことがもっとも重視されるシステムをミッションクリティカルシステムと呼ぶ。今日，多くのシステムがミッションクリティカルであり，この種のシステムを開発するには，当然のことながら要求された機能をいかに正確に実現するかが重要な課題となる。長年の技術蓄積により，ハードウェアの故障に起因する異常に関しては，この意味での信頼性を確保する技術はほぼ確立されつつある。しかし，ハードウェアやソフトウェアの設計エラーに起因する異常に対応する技術はその歴史の浅さからかまだ多くの課題を残している。そればかりか，システムの大規模・複雑化に伴い，今後ますます信頼性確保が難しくなるとの見かたもある。

この問題に対する一つの手立てとして，形式検証技術が注目されている。形式検証とは，システムが仕様どおりに正しく動作することを数学的方法によって保証する技術で，各種システムに対し高い精度の信頼性を保証するものとして期待されている。ここでは，制御ソフトウェアが仕様どおりに動作することを保証する際の有力な道具立てである形式検証技術について，その概要や原子力プラント制御への適用の試み，他分野での応用状況などを紹介する。

2 プログラムの形式検証技術

2.1 古くて新しい技術

プログラムの形式検証は古くて新しい技術である。基本的な理論は1960年代から研究され，1970年代にはバグのないプログラムをつくる技術として一躍脚光を浴びたが，実装における計算効率の問題などから，その後しばらくは“理論だけの技術”と位置づけられていた。ところが，理論面の整備，効率化技術の発達，計算機処理能力の向上などの技術的進歩とともに，現実のシステム開発における検証ニーズの高まりが背景となって，“必要に基づく技術”としての形式検証が，1990年代になって欧米を中心に盛んに研究され始めた[2]。当社でも，第五世代コンピュータ開発プロジェクト（FGCS）への参画を通じ，知的プログラミング開発支援システムMENDELS ZONE[3]やその検証支援系Metis-AS[4]などを開発し，ここで培われた検証の基本技術を現実のシステム開発へ適用するため，研究開発を続けている。

2.2 プログラムの正しさを数学的に証明

形式検証とは，ひと言で言えば，プログラムの正しさを証明する手法である。例えば整数の列を小さい順に並びかえるプログラムを考えたとき，普通のテストでは具体的なデータ“4, 1, 3, 2”を与えて，答えが“1, 2, 3, 4”となるのを種々のケースで確認する。これに対して，検証ではプログラムの正し

さとして「任意の整数列に対し，その要素を変えることなく，小さい順に並べ変える」といった命題を定め，プログラムがこの命題を満足するかを確認（証明）する。命題が証明されれば，これによって任意の入力データに対するプログラムの正しさが保証できる。テストによる確認では“2,1,1,2”のように同じ整数が重複して現れてもだいじょうぶか，“1,0,−2”のような0や負の整数に対しても正しく動作するかなど，すべてのケースを確認するのは不可能であり，未確認のケースがあると，その挙動は予測がつかなくなる。

形式検証ではまず，プログラムの正しさ，つまりプログラムの要求仕様を厳密に定義する必要がある。これを記述するために，形式仕様（Formal Specification）という数学的な仕様記述言語が用意されている。また，正しさの確認には，多くの場合に証明支援系というツールが用いられる（**図1**）。

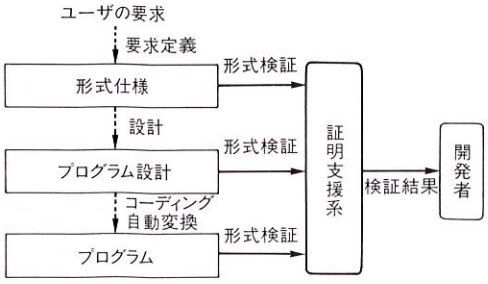

図1．形式検証を用いたプログラム開発の流れ　証明支援系を使って，形式仕様で書かれた要求に対する設計やプログラムの正しさを検証しながら開発を進める。

Program development using formal verification

2.3　形式検証の効果

上述の例で述べたように，形式検証を行うと任意の入力に対するプログラムの正しさが保証できる。一般に，プログラムの入力パターンには無限の組合せがあるので，形式検証の効果は非常に大きなものとなる。次に，形式検証の効果をまとめる。

(1)　仕様に対するプログラムの正しさを完全に保証できる。
(2)　形式仕様による仕様の明確化で，開発者間，ユーザ間の正確なコミュニケーションが促進される。
(3)　形式仕様自身が実行可能なため，仕様自身の正しさを早期に確認できる。
(4)　検証の経過や結果を，第三者が理解可能な信頼性保証の文書として残せる。
(5)　分野によっては，形式仕様からプログラムを自動生成する技術が確立されている。

形式仕様や証明支援系と言っても，唯一の言語やツールを指すわけではなく，応用分野ごとに最適の言語やツールが選択的に用いられている。そこで，以降では，形式検証の適用が進んでいるいくつかの分野について，用いられている技術の概要，適用状況，効果などを整理して紹介する。なお，混乱の恐れがない場合，形式検証のことを単に検証と呼ぶ。

3　原子力プラント制御への適用と評価

3.1　安全保護系制御のソフトウェア化

原子力プラントの安全保護系とは，原子炉を安全に制御・保護する安全管理システムである。近年，安全保護系の制御についてもソフトウェア化が進み，これに伴い，ソフトウェアの開発・検証に関する基準や指針が各国でまとめられている。これらの基準では，**図2**に示すような，検証と健全性確認（V & V：Verification and Validation）の実施が義務づけられている。ここで言う検証とは，各工程の上位と下位の記述（図書）の整合性を確認する作業であり，健全性確認とは，最終システムの動作が要求に合っているか否かを検査する作業である。検証の例としては，カナダのダーリントン（Darlington）発電所などが，安全保護系に関する要求仕様と設計仕様を表形式で記述し，両者を人手でつきあわせて整合性を確認するという検証を実施しているが，最近は標準化規格でも形式検証の導入を検討し始めている。当社でも安全保護系に対する形式検証の適用性に早くから着目し，現在のところ図2における第二検証，つまり，通常の要求仕様にあたる系統仕様書と設計仕様にあたるIBD（Interlock Block Diagram）の間の形式検証を実験的に成功させている。

3.2　プラント制御の論理検証方式

安全保護系の形式検証に適用したのは，当社が独自に開発したプラント制御の論理検証方式である。これは，次の流れで仕様が正しく設計に反映されていることを保証するものである[6]。

(1)　プラント制御の要求仕様を状態遷移図で記述する。
(2)　プラント制御の設計仕様をロジック図で記述する。

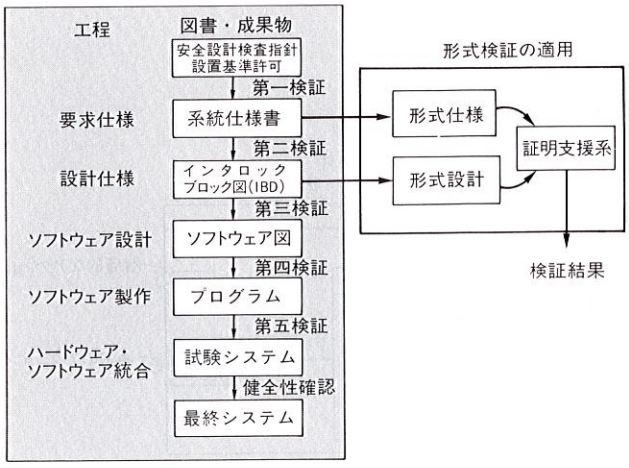

図2．安全保護系ソフトウェアの開発　系統仕様書とIBDの間（第二検証）の形式検証に実験的に成功している。

Software development for protection system

(3) 要求仕様と設計仕様の整合性を自動検証する。

つまり，安全保護系への要求仕様を状態遷移図で記述すれば，IBDによる設計仕様が要求仕様を完全に満たしていることを自動検証できる。状態遷移図は，プラントの挙動をわかりやすく表現する記述法として，各種制御の要求や設計の記述に広く用いられている。また，ロジック図はIBDとほぼ同等の記述法である。

3.3 形式検証の具体例

例として，**図3**のような注入弁の制御を考える。タンクの水位が低くなると自動的に注入弁を開いて水を注ぎ，水位が高くなると注入弁を閉じるという機構である。また，手動閉ボタンが押されると運転状況にかかわらず弁を閉じ，リセットボタンが押されるとすべてをリセットする。

図3の制御の要求仕様は，状態遷移図で**図4**のように記述できる。図では，箱が制御の状態を，矢印が状態間の遷移関係を表す。各状態には，状態名とその状態に遷移したとき実行するアクションが書かれている。アクションとして別の状態遷移図を書くことにより，階層的な制御構造も表現できる。遷移関係には，遷移の起きる条件が書かれ，黒い丸から遷移している先は制御の初期状態を指す。

これに対応するIBDの記述は**図5**のようになる。IBDは図の左が入力を，右が外部への出力を表し，その間を論理積や論理和などの記号で結んでいる。

これらの記述に対する形式検証の流れは，次のようになる。

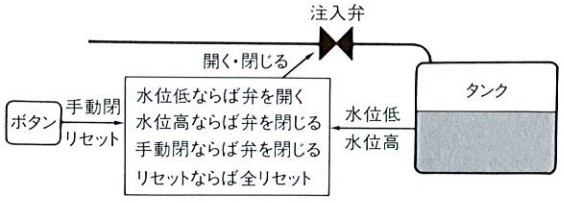

図3．タンク注入弁の制御システム　タンクの水位をセンサで感知し，注入弁の開閉を自動制御する。

Tank-valve control system

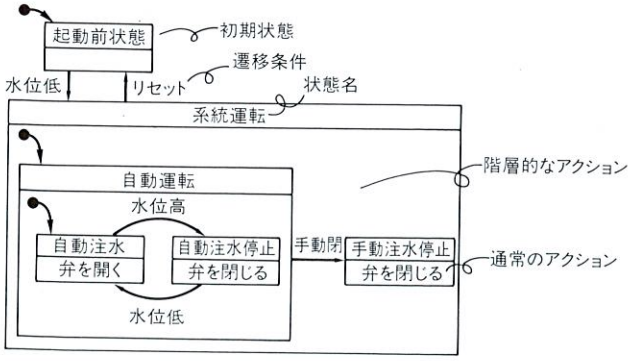

図4．状態遷移図の記述例　タンク注入弁の制御に関する要求仕様を状態遷移図で記述した例。

Example of state transition diagram

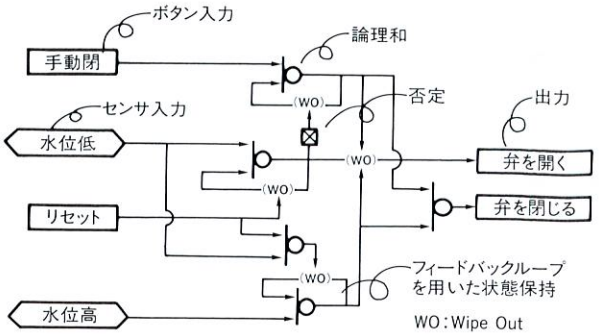

図5．IBDの記述例　タンク注入弁の制御に関する設計仕様をIBDで記述した例。

Example of IBD

(1) 状態遷移図の各状態に対応づけられるIBDの信号状態を求める。

(2) 状態遷移図の各状態におけるすべてのアクションが，IBDの対応する信号状態でも実施されることを確認する。

これによってすべての対応を調べ，両者に不整合のないことが確認されれば，「IBDが任意の入力信号に対して状態遷移図に書かれたとおりのアクションを行う設計である」ことが証明される。つまり，この意味での設計仕様の正しさが保証される。

3.4 形式検証の効果

現在，安全保護系の形式検証を行う実験システムAsksを試作中であり，この有効性確認のために検証実験を実施した。Asksは，状態遷移図エディタ，IBDエディタ，証明支援系から構成され，実験では2種類のエディタで記述した要求仕様と設計仕様の整合性を証明支援系によって自動検証することを試みた。この結果，以下の効果が確認された。

3.4.1 要求に対する設計の信頼性を厳密に保証する　実験において記述した要求仕様と設計仕様は，すべて証明支援系によって自動検証できた。一般に，要求と設計の間には情報のギャップが存在するため，この間の検証が自動化される効果は大きい。また，設計仕様からプログラムを自動生成する技術も研究している。

3.4.2 人手による検証工数を削減する　ここでの小さな例題についても，設計仕様の厳密な正しさを人手によって追求すると，論理の細かい記述にまで注意を払う必要があり，作業は思いのほか時間を要する。形式検証では，従来に比べて状態遷移図を記述する手間が増えるものの，検証作業の自動化によって，検証時間も含めた開発工数は大幅に削減できることが確認された。

3.4.3 不具合原因の特定が容易である　Asksにおいて設計の不具合が発見された場合，不十分な要求事項が状態遷移図に明示され，さらには，不具合を起こした際の信号状態がIBD上に表示されるので，開発者はこれらの情報を基に不具合の原因を特定することができる。

3.4.4 その他の効果　他の効果としては，要求仕様があいまいさのない形で表現されること，特に，処理の優先度や並行処理などの制御構造が，日本語記述よりも明確になる点が挙げられる。また，検証結果やその過程などを，第三者が理解可能な，信頼性の根拠を示す文書として残せる点も利点である。

4 他の分野における形式検証

4.1 シーケンス制御プログラムの検証技術

安全保護系では制御の論理検証を扱ったが，ここでは一連の制御の実行順序の正しさを検証する技術について紹介する。

4.1.1 モデル検査法による検証　シーケンス制御プログラムの検証システムSAVE/SFC[5]では，検証手法として時間論理を用いたモデル検査法を採用している。モデル検査法は次のステップから構成される。

(1) 検証対象であるプログラムから，そのすべての挙動を表すグローバル状態遷移グラフを生成する。

(2) グローバル状態遷移グラフのすべての状態をトレースし，時間論理式で記述された検証項目が満たされているか否かを検査する。

この検証法では，(1)のグローバル状態遷移グラフの状態数が爆発的に大きくなることがあり，これを回避する方法が主な課題となるが，すでにさまざまな回避方法を提案している。

4.1.2 化学プラントでの適用実験　実際の化学プラントを対象に検証実験を行い，システムの実用化に向けた評価を行った。この実験で生成されたグラフの状態数は，状態爆発を抑制する数々の縮約技術を適用した結果，約1,200状態であり，一つの項目の検証に要した時間は最長で約3時間であった。テストでは発見しにくい不具合も確実に検出され，検証の有効性が確認された。

4.2 家電システムの検証技術

広く一般のユーザが使用する家電製品は，あらゆる状況下での異常入力，誤操作などに対しても安全に作動することが求められる。当社では，家電製品向けCASE (Computer Aided Software Engineering) ツールIDSS (Integrated Development Support System)-μを開発しており，この中に安全性確保を主目的とした検証システムを実装している。

4.2.1 仕様検証と自動生成によるプログラム開発方式

IDSS-μは，製品の仕様を網羅的に検証するシステムと，仕様からソフトウェアを自動生成するシステムをもつ。これにより，仕様の信頼性確保と，コーディングミスによる不具合の排除が達成されている。

仕様検証システムは，状態遷移図で書かれた製品仕様と設計ルールが与えられると，ルールに違反する仕様上の誤りを検出する。設計ルールは，各製品の制御機器の操作やデータの状態に着目し，安全性・信頼性などの観点からハザード解析などを用いて作成される。主に次の項目を検出する[1]。

(1) 制御機器を誤った順序で操作する仕様

(2) 制御機器，データ間の関係に不都合を生じさせる仕様

(3) デッドロック状態に陥る仕様

(4) 実行不可能な範囲の仕様

4.2.2 IDSS-μにおける検証の効果　IDSS-μは，当社の冷蔵庫，電子レンジなどの家電製品の開発に実際に適用されている。この結果，検証システムの併用により，仕様上に含まれるバグの約80％以上を早期に発見，除去できることが確認された。また，約0.5時間程度で全仕様の検証が可能であり，開発の効率化にも寄与している。

5 あとがき

形式検証は難しいという意見を耳にすることがある。しかし，ここで紹介したように，形式検証のための記述・検証法は，扱いやすさの面からかなりくふうされてきており，一時期のような難しい数式を読書きすることもなくなった。それ以上に，今後ますます大規模・複雑化するソフトウェアに対し，テスト主体による信頼性確保は限界に近づいており，形式検証の導入は避けて通れないものとなりつつある。今後は，より扱いやすい検証技術の追求と，その実用化を目ざす。

文　献

(1) T. Fukaya, et al: Automatic Verifying Approach for Product Specification using FTA, 24th Inter. Sympo. on Fault-Tolerant Computing, pp.131-134 (1994)

(2) S. Gerhart, et al: Experience with Formal Methods in Critical Systems, IEEE Software, 11, 1, pp.21-28 (1994)

(3) 本位田真一，他：推論型システム記述言語MENDEL，情報処理学会論文誌，**27**，2，pp.219-227 (1986)

(4) 大須賀昭彦，他：代数的仕様を用いたソフトウェア開発支援環境：Metis-AS，情報処理学会論文誌，**36**，5，pp.1192-1202 (1995)

(5) 内平直志，他：シーケンス制御プログラムの検証技術，東芝レビュー，**48**，10，pp.775-778 (1993)

(6) T. Uraoka, et al: Automatic Verification of Consistency between Diagrams for Process Control, 2nd IFAC Workshop on Safety and Reliability in Emerging Control Technologies (1995)

大須賀　昭彦　Akihiko Ohsuga, D.Eng.

1981年入社。形式仕様化技術，検証技術などの研究に従事。現在，システム・ソフトウェア生産技術研究所主務，工博。Systems & Software Engineering Lab.

林　　俊文　Toshifumi Hayashi

1981年入社。原子力プラント運転支援システムの開発などに従事。現在，原子力技術研究所主務。Nuclear Engineering Lab.

神山　雅彦　Masahiko Kamiyama

1981年入社。原子力発電所制御システムの設計に従事。現在，原子力事業部原子力電気計装技術部主務。Nuclear Energy Div.