事例研究：組込みシステムの仕様検証とペアワイズ法を用いた評価項目の生成

A Case Study: Verification of Specifications of an Embedded System and Generation of Verification Items using Pairwise Testing

関澤 俊弦* 小鳥井 継**
Toshifusa Sekizawa, Tsugu Kotorii

組込みシステムでは信頼性の確保が極めて重要である。信頼性は様々な形式手法を用いて確保できる。我々は開発製品に形式手法を適用して信頼性を検証する技術について研究している。本稿では実際の開発に形式手法を適用した事例を紹介し、そのアプローチと検証結果について説明する。本研究の目的は評価項目と呼ぶパラメータに対する制約を含む仕様の正当性を確認することであり、そのためにモデル検査と充足可能性判定を採用して検証を行った。本研究では仕様から形式モデルへの変換ルールを設定したが、変換の一部は手動で行った。手動による変換は評価項目の生成を制限するため、ペアワイズ法を用いた検証パラメータの自動生成を試みた。最後に検証結果を提示する。形式手法の適用は現時点ではまだ予備段階にあり、製品の自動的な検証に向けて形式手法を発展させていきたいと考えている。

キーワード：組込みシステム、検証、モデル検査、検証項目の生成、ケーススタディ
※本論文はAPSEC2013で採択され発表した原稿をIEEEの許諾の下で翻訳し、掲載するものである。

Ensuring the reliability of embedded systems has become very important. Reliability may be ensured by a number of formal methods. We study one such verification technique by applying it to an in-house development product in Mitsubishi Space Software Co., Ltd. This is a practical industrial case study, we describe our approaches and present verification results. Our aim is to check the correctness of specifications which include a set of constraints on parameters individually called an evaluation item. To that end, we adopt model checking and satisfiability checking. In our study, we set conversion rules from specifications to formal models. Part of the conversion is done by hand in this study. Manual generation limits the preparation of individual evaluation items. To overcome this limitation we present an approach for automatically generating combinations of parameters for verification by applying the pairwise testing method. Finally, we present experimental results. Note that the application of formal techniques, in this setting, is still in its preliminary stages. It is intended to develop formal techniques to the point where products may be automatically verified.

Keywords-embedded system, verification, model checking, generation of verification items, case study

1. まえがき

組込みシステムは現代社会において重要な位置を占めている。近年、高い安全性が求められるシステムから家庭用電化製品に至るまでほぼ全てのものがソフトウェアにより制御されるようになったため、組込みシステムの開発では設計仕様を検証する重要性がより一層高まっている。形式手法は数学を基礎とした検証・開発技術であり、設計仕様を検証するために広く用いられている。成功事例として、プロトコル仕様[1]、グループウェア・システム[2]、航空電子機器による滑走路監視[3]への適用例などが報告されている。形式手法と準形式手法のこうした状況は、ISO/IEC 61508やRTCA（航空無線技術委員会）DO-178Cといった国際規格での採用につながっている。そのため、製品開発プロセスに形式手法を導入することで、こうした国際規格へ対応することが重要である。

＊大阪学院大学（現在、日本大学） ＊＊関西事業部 品質保証部

モデル検査[4]は多くのシステム検証に適用され成功を収めた形式手法の1つであり、モデルの状態空間を網羅的に探索することにより設計仕様を満足するか検証する手法である。本手法ではモデルはクリプキ構造等の状態遷移系で表現され、検査式（検証される性質）は線形時相論理（LTL）や計算木論理（CTL）[5]等の論理式で記述される。

本稿では、組込み制御システムの開発に形式手法を適用した事例について報告する。製品開発プロセスへのモデル検査の適用はまだ試験的な段階であるため、本研究は開発プロセスへの形式手法、特にモデル検査の適用可能性に関する調査も含む。本目的のため、主な機能が2つの仕様記述（機能仕様と信号の組合せ表）で定義される組込みシステムを選択し、以下の3つのアプローチを試みた。

1）モデル検査を用いた到達可能性判定
2）SMT（充足可能性モジュロ理論）ソルバを用いた充足可能性判定
3）ペアワイズ法を用いた検証用変数の組合せの生成

アプローチ1）では機能仕様から状態遷移システムを構築し、組合せ評価表から到達可能性の論理式を導いた後、モデル検査を用いて検証を行った。アプローチ2）は状態遷移なしとした検査式にのみ適用する、一種の有界モデル検査である。アプローチ3）は、アプローチ1）や2）と異なり、手動で作成した組合せ表から生成した検査式を検証している。手動で作成できる組合せの数には限度があるため、組合せの生成にはペアワイズ法を用いた。

実験を通して、レビューを経た仕様の記述に不十分な点が含まれていることを確認した。本研究の成果は、i）製品仕様を形式モデルに変換したこと、ii）実用化の事例研究として検証結果を示したこと、iii）ペアワイズ法により検証用パラメータの組合せが生成できることを示したこと、である。なお、本稿では企業秘密保護のためシステム名や変数名は一般的な用語で記述している。

本稿の構成は次のとおりである。２章ではシステムと仕様について記述する。３章では対象システムと仕様の正しさを調査するための、モデル検査の適用方法を示す。４章では検証結果を示し、５章では検証結果について考察する。６章ではまとめと今後の課題について説明する。

２．システムと仕様

本研究における対象システムの振舞いを次の仕様で定義する。

1）制御システムの機能仕様
2）状態フローに関連する信号の組合せ表

仕様1）に関して、機能仕様は組込み製品開発の中核を成す。この機能仕様をそのまま基盤として使用したり拡張を加えたりしながら多くのプロジェクトを進めるため、機能仕様の検証を保証することは極めて重要となる。留意すべき点は、仕様についてレビューが実施されているが、さらなる信頼性の検証が求められることである。

機能仕様では、信号（変数）、プロセス、および各システム状態での遷移条件を定義している。信号については、正常信号とエラー信号を含む15個のbool値信号σ_1、σ_2、...、σ_{15}が定義されており、各信号は対象システムのセンサの状態を表す。対象システムは信号の値に依存する7つのシステム状態S_1、...、S_7を取り、各システム状態は「初期診断」のような状態で表される。遷移条件もまた機能仕様で定義される。表１に遷移条件の一部を示す。

表１　システムの遷移条件

state	No.	condition	transition
S_1	#S1-1	AND {(σ_1=ON), (σ_5=OFF)}	S_2
S_2	#S2-1	OR {(σ_1=OFF), (σ_5=ON)}	S_5
:	:	:	:
S_7	#S7-1	OR {(σ_1=OFF), (σ_2=OFF), (σ_3=ON)}	S_5
S_7	#S7-3	AND {(σ_3=ON), (σ_2=ON)}	S_3

対象システムには遷移条件が16個存在する。表１には、現状態（state）、条件番号（No.）、遷移条件（condition）、および遷移先（transition）を列記している。例えば、最初の行は、現状態がS_1、条件番号が#S1-1、すなわち、遷移条件が（σ_1＝ON）かつ（σ_5＝OFF）が成り立つ時、システム状態はS_2に遷移することを意味する。

機能仕様ではシステム全体を以下のP_1、...、P_4の4つのプロセスで構成する。

1）P_1：メインコントローラ
2）P_2：人の操作
3）P_3：外部環境
4）P_4：P_3以外の外部環境

これら4つのプロセスは並列処理され、P_2以外のプロセスは常に動作している。プロセスP_1は表１の遷移条件で定義される振舞いに従ってシステム状態を取り扱うメインコントローラである。このメインコントロールプロセスが検証対象である。プロセスP_2は人の操作を模倣し、システムのメインスイッチを表すグローバル信号σ_1のみを扱う。メインスイッチは電源供給に割り込みをかけず、メインスイッチがオフにされても全てのプロセ

スは個々に動作を続ける。プロセスP_3は外部環境を表し、センサ値の変化を検出・収集してP_1に通知する。プロセスP_4も外部環境であり、P_3と同様の動作を行う。

ここで、プロセス間のデーターフローについて説明する。プロセスP_2はシステム状態に影響を及ぼすグローバルスイッチσ_1を制御する。例えば、σ_1がOFFの場合σ_2は決してONにならないが、σ_1がONでかつ他の条件が成り立つ場合σ_2はONになる。しかし、ハードウェアセンサはたとえσ_1がOFFでも仕様外の条件により動作する。ここでは組込みシステムの中核部分のみを扱っており、センサは外部の条件に影響を受けるからである。プロセスP_3とP_4はセンサの変化を検出し、P_1に非同期的にセンサ値を送信する。メインコントローラP_1は定期的なポーリングによりこれらのセンサ値を収集し、センサ値の組合せに従い遷移条件が成り立つかどうか判断する。現在の状態の遷移条件が満たされれば、P_1は他のシステム状態への遷移を行う。

仕様2）に関して、評価表には35個の評価項目を含む。ここで、評価項目とは15個の信号σ_1、…、σ_{15}の組合せを、評価表とは評価項目のリストを意味する。評価項目はレビュー結果から手作業で抽出されたため、P_1の信頼性を確保するために不可欠な信号の組合せであると言える。なお、評価表は試験段階で試験項目を作成する際に用いられる。

表２に35個の評価項目を含む評価表の一部を示す。最初の3行は、プロセス番号、信号名、および信号のタイプ（正常（N）またはエラー（E））を表す。4行目以降のデータ行は評価項目（項目番号、各信号の制約、現在の状態、期待される遷移）を表す。信号の制約のセルは、値が入るか、もしくはブランクとなる。セルに制約の値が入る場合信号はその値を取り、ブランクの場合信号が取り得る範囲で非決定的な値となる。状態（state）列は仕様1）で定義された現在のシステム状態を表す。遷移（transition）列は期待される検査式を表す。「$S_i \rightarrow S_j$」と記述されている場合は状態S_iから状態S_jへ遷移することを意味し、「現状維持」(Status Quo）の場合は状態遷移が起こらず現在の状態に留まることを意味する。例えば、評価項目No.1はプロセスP_2で正常信号σ_1＝ON、プロセスP_4でエラー信号σ_5＝OFFであり、それ以外の全ての信号は非決定的にONかOFFのいずれかの値を取ることを示す。現在のシステム状態がS_1である場合、対象プロセスP_1が状態S_2へ遷移を行うことが期待される。

3. アプローチ

本研究のアプローチは、評価表により定義される検査式の検証、およびペアワイズ法を用いた評価項目の生成、の2部で構成される。本アプローチを図１に示すとともに、以下に詳細を説明する。

A. 検査式の検証

検査式の検証はモデル検査器SPIN[6]を用いて、充足可能性判定はSMTソルバYices[7]を用いて実施する。これらの手順を以下に詳述する。

1）モデル検査

評価項目で「$S_i \rightarrow S_j$」と記述される到達可能性の検証にはモデル検査器SPINを使用する。SPINでは仕様記述言語Promelaで記述したモデルとLTLで記述した検査式を用いる。評価表には多くの評価項目が含まれるが、それらは互いに矛盾する場合もあるため、逐一検証することにする。

表２　評価表

	P_2	P_3			P_4												
No.	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7	σ_8	σ_9	σ_{10}	σ_{11}	σ_{12}	σ_{13}	σ_{14}	σ_{15}	state	transition
Type	N	N	N	E	E	N	N	N	N	N	N	N	E	E	N		
1	ON				OFF											S_1	$S_1 \rightarrow S_2$
2	OFF				ON											S_1	Status Quo
:	:																:
21	ON	ON			OFF									ON		S_4	$S_4 \rightarrow S_6$
22	ON	ON			OFF								ON			S_4	$S_4 \rightarrow S_7$
:	:																:
31	ON	ON			OFF									ON		S_6	Status Quo
:	:																:
35	ON	ON			OFF				ON				OFF	OFF		S_7	$S_7 \rightarrow S_3$

Promela入力モデルは、変数宣言、4つのプロセス、およびいくつかのインライン関数から構成され、仕様から直接的な方法で作成される。本研究では検証の自動化を見据え、仕様からPromelaへの変換ルールを策定した。策定した変換ルールは主として、遷移条件からプロセスP_1を生成するルール、および評価項目から他のプロセスを生成するルール、の2つからなる。

まず、仕様1）から変換するルールを説明する。表１に示すように、システムの遷移条件は信号を原子命題とした述語論理で表現されるため、Promelaモデルへの変換は容易である。例えば、条件#S1-1は図２に示すPromelaモデルに変換される。ここで、プロセスP_1は検証対象であり全ての検証で共有されるため、この変換は一度のみ行われる。

次に、評価項目からPromelaへの変換について説明する。変換は変換ルールに従って自動的に行われる。プロセスP_2は1つのグローバル変数のみを扱うため、図３に示すように容易にモデル化できる。プロセスP_3およびP_4は基本的に同じ動作をする。2つのプロセスに分ける理由は対象システムの設計に基づくものであり、本研究の範囲外である。両プロセスともP_1に信号を送信し続けて停止しないため、プロセス全体は無限ループとして表現される。1つのdoループの実行は1回の信号の送信に相当する。信号の値は評価項目の要素の値に依存する。制約によって定まる場合は決定的に表され、そうでない場合は非決定的に選択される。図４にPromelaコードで記述した変換ルールを示す。決定的な信号がない場合には信号の変換がスキップされる点に注意が必要である。

さらに、検証には少なくとも１つの検査式が必要となる。この検査式は遷移先の状態と期待される検査式が書かれた検査項目のセルから抽出される。期待される検査式が「$S_i \rightarrow S_j$」の場合、状態S_iからいずれS_jに達する到達可能性を意味し、□（state＝S_i→◇（state＝S_j））というLTL式に変換される。ここで、stateはシステム状態を表すPromelaの変数である。

2）充足可能性判定

評価項目の遷移検査式が「現状維持」(Status Quo）の場合は状態遷移が起こらない。

本節では「現状維持」である評価項目に対するアプローチについて説明する。この期待される検査式を対応するLTL式で記述することは可能であるが、信号数により状態爆発問題が発生してしまうため、SPINを用いた検証は困難である。先に述べたようにプロセスP_1の遷移条件は述語論理式で表され、評価項目内の信号に対する制約も述語論理で表現可能であるため、充足可能性判定によって遷移条件の判定を行うことができる。

具体的には、まずシステム状態S_iに関連する遷移条件φと評価項目によって定義される制約χを得る。次に制約χの下でのφの充足可能性を判定する。φが充足不能な場合、状態S_iから状態遷移する信号の組合せは存在しない。φが充足可能な場合、少なくとも1つの信号の組合せが存在するため、「現状維持」という期待値と反することになる。この場合Yicesは判定された論理式を満たす信号の組合せを示し、これは反例となる。

本アプローチは、境界値が現在の状態に制限される有界モデル検査[8]の1種である。全ての信号が2値のため検査式の検証にはSATソルバで十分であったが、将来の仕様拡張版では信号の範囲に実数や関数を含む可能性があるため、本研究ではSMTソルバを採用した。

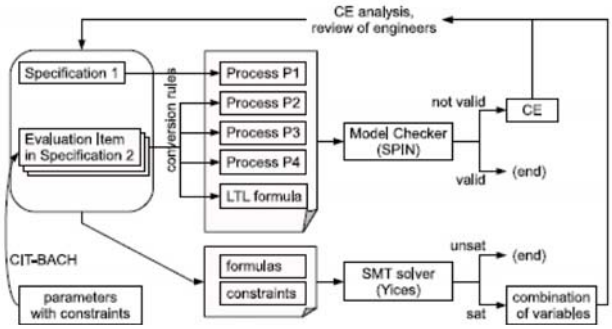

図１　アプローチ

```
if
  state==S1 ->
    if
    ::(s1==true&&s5==false)->atomic{state=S2}
    ::else->skip
    fi;
```

図２　遷移条件の変換例

```
proctype P2() {
  (s1 == false) -> s1 true;
}
```

図３　プロセスP_2の変換ルール

```
proctype processname(){
  do
  :: true ->
    (constraint)*
    (if
    (:: non-deterministic choice)*
    fi;)?
  od
}
```

図４　評価項目の変換ルール

B. 評価項目の生成

評価項目を手動で作成していては検証できる検査式の数が限られてしまう。評価項目の自動生成が必要となるため、パラメータのペアから試験項目を生成する組合せ手法であるペアワイズ法を採用した。すなわち、ペアワイズ法を用いて評価項目を生成し、モデル検査により検査式の検証を行った。ペアワイズ法はパラメータのペアを生成するのみであるため、検査式を各評価項目に関連付けることが必要である。ペアワイズ法により評価項目を生成するためにCIT-BACH（BDDを用いた制約処理機能を備えた組合せ試験項目生成ツール）[9]を使用した。CIT-BACHの特徴の1つは、パラメータへの制約の適用を可能にすることである。すなわち、入力パラメータに制約を与えれば、生成されたパラメータのペアはこれらの制約を満たすことになる。

ここで、CIT-BACHの結果を用いたモデルの構築について検討する。仕様1）に基づくSPIN入力モデル、および評価項目の制約となる信号の組合せを生成することは可能であるが、CIT-BACHは信号の組合せを生成するのみであり検査式の生成は行わない。本問題を解決するため、システム状態を正常状態とエラー状態に分類できることに着目した。システム状態S_5、S_6およびS_7はエラー状態に割り当てられており、仕様によれば、少なくとも1つのエラー信号がONの場合、システムはエラー状態の１つに遷移しなければならない。この遷移は間接的な遷移であり、例えばS_1は直接エラー状態の1つに到達することはできず、いくつかの他の状態を経て間接的に遷移する必要がある。この仕様は検査式の設定に適用できる。具体的には、ペアワイズ法の入力パラメータの中で少なくとも１つのエラー信号をONに設定し、検査式をS_5、S_6またはS_7への到達可能性判定とする。

本研究では上記のアプローチを手作業で行っているが、本アプローチは自動化が可能であり、計算機環境の空き時間に検証できるようになる。このように検証を重ねることが信頼性向上に寄与すると考える。

4. 検証結果

本章では、検証と充足可能性判定の結果について説明する。３章で説明した変換ルールに基づき、評価表から入力ファイルを生成するためのプロトタイプツールを実装した。プロトタイプツールはSPIN用入力ファイルを生成するのみでありYices用入力ファイルは手作業で作成しているが、Yices用入カファイルはそれほど複雑ではなく、困難はない。

仕様2で示した35個の評価項目をすべて検証した。このうち25項目をSPINで、10項目をYicesで検証した。その結果、3つの評価項目No.21、No.22およびNo.31に関して、遷移条件を満足するには十分でないことが判明した。このうちNo.21とNo.22はモデル検査によって、No.31は充足可能性判定によって発見された。仕様はレビューされていることから全ての検査式が成り立つと予想しており、検証結果は想定外であった。

検証結果に対して誤りの原因を調査した。まず、変換ルール、特にプロセスP_1とそれに関連する変換ルールを詳細に調査した。これは、プロセスP_1は一度だけ生成され全ての検証で共有されており、P_1に不具合があれば検証結果が意味をなさないためである。調査の結果、プロセスP_1は仕様を正確に反映していると結論づけた。以降ではプロセスP_1は正しいと仮定する。

次に、評価項目No.21およびNo.22における遷移条件を調査した。評価項目No.21では、期待される遷移検査式は到達可能性：□（(state＝S_4）→◇（state＝S_6）である。反例を分析したところ、必要な制約が欠落していることが判明した。製品開発を行うエンジニアの視点を持ち合わせておらず品質保証の観点からのみ行ったものであったため、製品担当エンジニアに検証結果のレビューを依頼したところ、やはり制約の欠落が確認された。具体的には、評価項目No.21において信号σ_{13}は非決定論的な変数として定義されていたが、実際にはOFFでなければならなかった。信号σ_{13}をOFFに設定し検査式を再度検証したところ、成り立つことが確認された。評価項目No.22の問題はNo.21と同様に制約の欠落であり、適切な制約が設定された後は満足のいく結果になることを確認した。なお、これらの再検証では再びプロトタイプ変換ツールを使用した。さらに、本製品では評価項目は試験項目を作成する際のベースとなっていたことから、これらの問題の影響について調査を行ったところ、これらの制約の欠落は試験段階では暗黙の了解として扱われており、従って不具合は回避されていた。言うまでもなく、このような事象はソフトウェア開発における信頼性を低下させる。

続いて、評価項目No.31における問題を調査した。「充足不能」となる結果を予想したが、SMTソルバは「充足可能」を示した。Yicesが返した充足可能な遷移条件を満たす変数の組合せについて分析したところ、変数間の矛盾が特定され、評価項目No.31の条件が十分ではないとの結論に達した。評価項目No.21、No.22の場合と同様エンジニアに分析結果の確認を依頼した結果、評価項目No.31は形式手法の効果を検証するためエンジニアによって意図的に評価表に挿入されたものであったことが判明した。エンジニアによれば、評価項目No.31は変数および制約が誤って記述されている状況を模倣するもので

あった。評価項目No.31は実際の仕様には存在せず、意図的に追加されたことは通知されていなかった。評価項目No.31は実験的な項目ではあるものの、この矛盾点が検出できたことは充足可能性判定の適用可能性を示していると考えられる。

最後に、CIT-BACHを使用して試験的に評価項目を生成した検証結果について説明する。有効性を確認するため、３章で示したアプローチに従い、CIT-BACHを使用して評価項目の組合せを生成した。本アプローチでは制約となるエラー信号の設定が求められるため、ここではエラー信号σ_5をONに設定した。また正常信号σ_1をOFFにすると他の多くの信号に影響を及ぼすため、σ_1もONに設定した。これらの制約からCIT-BACH用入力ファイルを作成し実行させたところ、CIT-BACHは2個の固定信号と13個の可変信号から9個の評価項目を生成した。さらに、検証にはシステムのターゲット状態と少なくとも1つの検査式の決定が求められるため、ここではターゲット状態をS_3、検査式を「エラー状態のうちの1つに到達可能」とした。到達可能性検査式は□(*targetstate*→◇*errorstates*) であり、*targetstate*は「state＝S_3」、*errorstates*は「state＝S_5 ∨ state＝S_6 ∨ state＝S_7」と表される。エラー状態が3つのシステム状態で構成される理由は、制約のためエラー状態を特定することができないためである。プロトタイプ変換ツールを使用して生成した評価項目をPromelaモデルに変換し、SPINを使用して検証を行った。その結果、想定どおり矛盾は見つからなかった。なお、上記において全ての信号に固定値を設定している。もちろん一部の信号に非決定的な値を設定することは可能である。

検証にはCPUがIntel Core i5-2400 3.10GHz、メモリが8GBのPCを使用し、Windows 7 64 ビットOS上でSPIN 6.2.2、Yices 1.0.38、およびCIT-BACH 1.01を使用して行った。SPINの検証ログによると保存されたシステム状態数はおよそ3×10^7であり、検査式が成り立つケースにおける検証時間はおよそ450秒であった。未使用変数を省略したため、Yicesは約1秒で結果を返した。

５．考察

本章では、本研究での検証実験について検討するとともに、制御工学や組込みシステム分野における関連研究について考察する。

A. 検証実験に関する検討

本節では、検証実験について検討し、開発プロセスへのアプローチの有効性について考察する。

まず、評価項目における必要な信号値の欠落について検討する。先に述べたように、意図的に挿入された試験的な評価項目は含まれていたものの、評価表はレビューされていた。信号値の欠落による不具合の発生が暗黙の了解によって避けられるとしても、評価項目が試験項目のベースとなることから、設計段階でそれらを検出できることは製品開発において有益である。意図的に挿入された評価項目を検出できたこともまた形式手法の可能性を示している。

CIT-BACHを使用して生成した評価項目の検証では何らの矛盾を発見することはできなかった。だとしても、これらの評価項目は新しい信号の組合せを指定するものであり、レビュー段階で抽出されなかった項目の検証を可能とする。この結果は本研究で試みたアプローチの適用可能性と有効性を示している。本研究の目的の1つは形式手法の適用可能性の調査であった。形式手法は開発上流工程における問題軽減に効果があると言われているが、形式手法は有用でありこの説は妥当な見解であることが本研究結果により裏付けられた。

有用な結果が得られた一方で克服すべき問題も残っており、その1つが自動化に関する問題である。形式手法を製品開発へ適用する場合、自動化は重要な検討事項である。本研究では形式モデルを生成する際に変換ルールを適用したが、本アプローチが自動化を可能にすると考えられる。一方で反例は手動で解析しており、形式手法の活用にとって望ましい状況ではない。これは、反例は必ずしも最短経路を示すものではなく、また手動による解析には技術とコストが必要になるためである。それ故に、計算機による自動化された反例解析が望まれる。もう1つの問題は状態爆発問題である。状態爆発問題はモデル検査における重大な課題と考えられている。本研究ではモデルを仕様から直接的な方法で構築したが、本アプローチではモデルの可読性は向上するものの状態爆発問題を引き起こす。本研究における状態空間サイズはSPINの許容範囲内ではあったがほぼ限界値に達していたことから、本アプローチを製品仕様の拡張版に適用する場合、状態爆発は潜在的に発生する問題となり得る。抽象化は状態空間の大きさの削減に向けた有望な技術と考えられ、特にデータマッピングおよび述語抽象化は効率的と考えられる。

次に、機能の拡張性について検討する。従来のモデル検査のため仕様をPromelaモデルに変換した。全ての信号が2値のため本研究で扱う仕様には本アプローチで十分であったが、仕様は本研究における対象製品の中核を成しており、拡張版では信号は2値に限定されない。さらに、様々な種類の検査式が検証対象となることが予想される。例えば、組込みシステムの振舞いに関する重要な側面の1つに、時間に関連した性質がある。このよう

な実時間システムを扱うためにいくつかのモデルが提案されており、その1つに時間オートマトン[10]がある。時間オートマトンは、時間に関連した検査式が検証できるように今回試みたアプローチを拡張する際の候補になると考えられる。他には、連続ダイナミクスと離散ダイナミクスが時間進行とともに混合し合う、ハイブリッドシステム[11]の検証がある。組込みシステムは連続系を制御することもあることから、ハイブリッドシステムもまた対象の構造をより正確に再現するモデルになると考えられる。

B. 関連研究

本節では、制御工学における変換や適用の検証に関連する研究について説明する。ソフトウェア製品の品質確保を目的として形式手法技術の適用を検討する際には、モデル構築と検査式の記述について考慮する必要がある。これまで仕様から形式モデルへの変換について数多くの研究が成されており、例えばリアルタイム制御プログラムの検証[12]では、制御プログラムから検査式への自動変換を用いて時間オートマトンモデルを構築したことが報告されている。また、SysMLやUML等の仕様記述言語からモデル検査で使用される形式モデルへの自動変換を行った事例も報告されている[13]。いくつかの検証技術を要約して安全性と活性の検証結果を記述した、ロボティクス制御システムのモデル検査に関する調査報告[14]も参考になる。学術分野から産業的実用化への技術移転プロセスについて記述した論文[15]もある。

6. 結論

我々は、自社開発製品である組込みシステムに関する仕様の信頼性を確保するための検証技術について研究した。本研究の目的の1つは形式手法の適用可能性の調査であった。本目的のため、モデル検査や充足可能性判定を目的とした仕様から形式モデルへの変換、およびペアワイズ法を用いたパラメータ組合せの生成、の2つのアプローチを試みた。まず仕様を形式モデルに変換する変換ルールを設定し、次に変換ルールの適用により対象システムのモデルを構築し、モデル検査器SPINやSMTソルバYicesを使用して検証を行った。その結果、仕様のパラメータ記述に不備があることを発見した。さらに、ペアワイズ法を用いた評価項目生成のアプローチを示した。本アプローチはレビューで抽出されなかった項目の検証を可能にし、仕様の信頼性向上に寄与する。これらの結果は、形式手法の有用性を強く確信させるものである。

本研究により形式手法の適用可能性は有望であることが示されたと考えられるが、同時にいくつかの課題も明らかになった。その1つは、本研究で試みたアプローチの実装である。本研究では、検証や充足可能性判定に用いる入力ファイルを生成するためにプロトタイプ変換ツールを使用した。ツールは入力ファイルの一部を変換するのみであり、詳細部分は手動で記述する必要があったが、全ての変換は機械的に行われた。これは評価表からの変換が完全に自動化できることを示すものであり、開発現場や製品への実用的な展開が望ましい効果を生み出すと確信させる結果である。もう1つの課題は、B、Z、およびVDMのような形式仕様言語の導入である。本研究では、仕様が自然言語で記述されていたためメインコントローラを手動で変換する必要があった。当然、このような仕様記述は形式手法を用いた検証の自動化を困難にする。

謝辞

ペアワイズ法とツールCIT-BACHについて助言を頂いた、大阪大学の土屋達弘教授に深謝の意を表します。

本論文について校正の労を頂いた、大阪学院大学のR. D. Logie准教授に感謝の意を表します。

参考文献

[1] A. Schneider, T. Bluhm, T. Renner, U. Heinkel, J. Knablein, and R. Zavala, “Formal verification of abstract system and protocol specifications,” 2012 35th Annual IEEE Software Engineering Workshop, vol. 0, pp. 207–211, 2006.

[2] M. H. ter Beek, M. Massink, D. Latella, S. Gnesi, A. Forghieri, and M. Sebastianis, “A case study on the automated verification of groupware protocols,” in 27th International Conference on Software Engineering (ICSE 2005), 15-21 May 2005, St. Louis, Missouri, USA, G.-C. Roman, W. G. Griswold, and B. Nuseibeh, Eds. ACM, 2005, pp. 596–603.

[3] R. I. Siminiceanu and G. Ciardo, Symbolic Model Checking for Avionics. John Wiley & Sons, Inc., 2012, pp. 85–112.

[4] E. M. Clarke, O. Grumberg, and D. Peled, Model Checking. MIT Press, 1999.

[5] E. A. Emerson, “Temporal and modal logic,” in Handbook of Theoretical Computer Science. Elsevier, 1990, vol. B, ch. 16, pp. 995–1072.

[6] G. J. Holzmann, The Spin Model Checker. Addison-Wesley.

[7] “The Yices SMT Solver,” http://yices.csl.sri.com/.

[8] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, “Symbolic model checking without BDDs,” pp. 193–207.

[9]“CIT-BACH Website,” http://www-ise4.ist.osaka-u.ac.jp/～t-tutiya/CIT/.
[10] R. Alur and D. L. Dill, “A theory of timed automata,” Theoretical Computer Science, vol. 126, no. 2, pp. 183–235, 1994.
[11] J. Lunze and F. Lamnabhi-lagarrigue, Handbook of Hybrid Systems Control - Theory, Tools, Applications. Cambridge University Press, 2009.
[12] T. K. Iversen, K. J. Kristoffersen, K. G. Larsen, M. Laursen, R. G. Madsen, S. K. Mortensen, P. Pettersson, and C. B. Thomasen, “Modelchecking real-time control programs: verifying lego(R) mindstormsTM systems using UPPAAL,” in 12th Euromicro Conference on Real-Time Systems. IEEE Computer Society, 2000, pp. 147–155.
[13] L. Alawneh, M. Debbabi, Y. Jarraya, A. Soeanu, and F. Hassaïne, “A unified approach for verification and validation of systems and software engineering models,” in 13th Engineering of Computer Based Systems. IEEE Computer Society, 2006, pp. 409–418.
[14] N. Sharygina, J. C. Browne, F. Xie, R. P. Kurshan, and V. Levin, “Lessons learned from model checking a NASA robot controller,” Formal Methods in System Design, vol. 25, no. 2-3, pp. 241–270, 2004.
[15] R. P. Kurshan, “Verification technology transfer,” in 25 Years of Model Checking, ser. Lecture Notes in Computer Science, O. Grumberg and H. Veith, Eds., vol. 5000. Springer, 2008, pp. 46–64