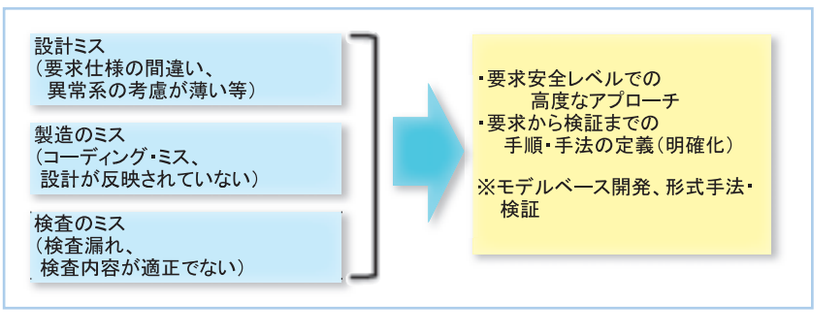
状態遷移表に形式検証を適用する“嬉しさ”

片平　典幸
福岡県産業・科学技術振興財団
イノベーションクラスター戦略本部
プロジェクトリーダー

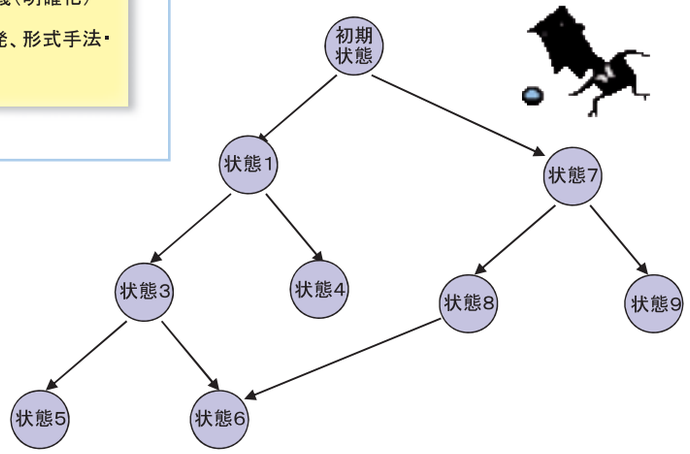
■はじめに

近年、電子機器に関するIEC61508や自動車のISO26262などで、機能安全の必要性が強く求められています。機能安全は、自動車以外にも産業機械, 鉄道システム, 航空機, 医療機器など他の分野にも専用の規格が準備されており適用範囲は広がる一方です。では、機能安全とは何でしょうか？　安全な製品を開発するために有効と考えられる管理や手法を定めたものを指します。

それは、今後、新しく提供されていくであろう機能・サービスに対しても同様に適用されていくと考えられます。新しいエネルギー源である水素の活用や人間に近い場所に存在する接触型ロボットなどでも機能安全の確保が要求されることとなるでしょう。また、それらのものには組込みソフトウェアでの制御が必須となります。組込みソフトウェアは我々の生活の中に全く意識することなく、空気のように存在しています。例えばAV機器、例えば車、例えばあなたの乗ってきたエレベータ。ユーザーにとっては全くその存在を意識することのない組込みソフトウェアは、今まで以上に安全に存在している事が求められているのです。そういったソフトウェアの機能安全を確保するのに使われる検証方法が形式検証（モデル検査）なのです。形式検証を要求仕様に適用するならば、要件が満たされるという意味で妥当性確認の一つと考えることも出来ます。我々は状態遷移表を要求仕様が記述できる状態遷移モデルとして形式検証を行うという研究開発を行っています。そのことの、何が嬉しいのかについて、我々の研究内容を通して述べていきたいと思います。



［図1］　ソフトウェアの機能安全

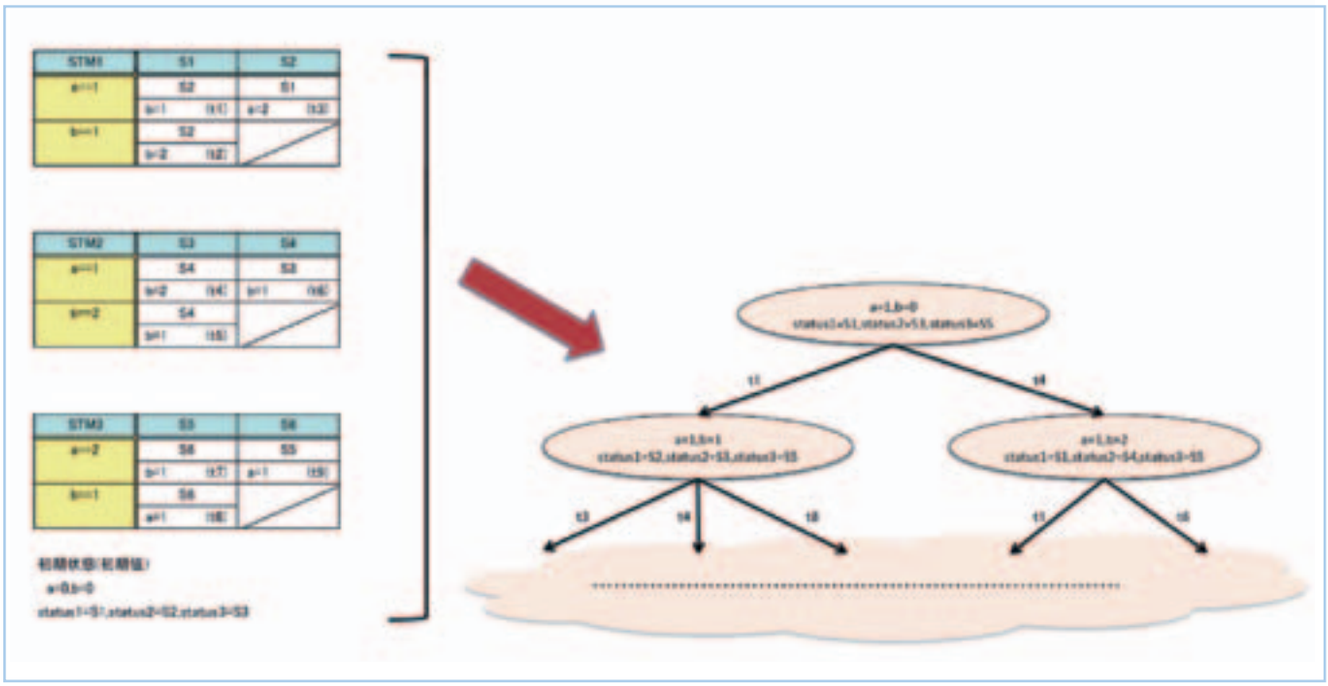


［図2］　モデル検査の網羅的検査イメージ

1.形式検証とは

形式検証とは、形式的な記述仕様に基づいて記述されたものを数学的に証明することを言います。その一つとしてモデル検査があります。モデル検査とは、ある特定の性質に対して取りうる全状態を自動的・網羅的に検査する手法です。

では何故、形式検証が必要なのでしょうか？　今までのソフトウェア試験（単体テスト・総合テスト等）ではダメなのでしょうか？　形式検証で検証する対象は状態遷移モデルです。　我々は形式検証の対象をソースコードではなく、状態遷移表で書かれる要求仕様としています。要求仕様時の検証は、今までソフトウェアの世界では、レビューという人為的な検証は存在していますが、システマイズされた検証は、一般的に行われてはいません。要求仕様とは“考え方”であり、根本になる“考え方”に存在している間違いを上流開発で見つけることができれば、より安全な製品・サービスを提供することが出来ます。そのため、組込みソフトウェアの世界でも機能安全を確保するために形式検証による安全性確保が求められるのです。



［図3］　並列状態の状態遷移表とそのモデル検査時の状態遷移（クリプ木構造）関係

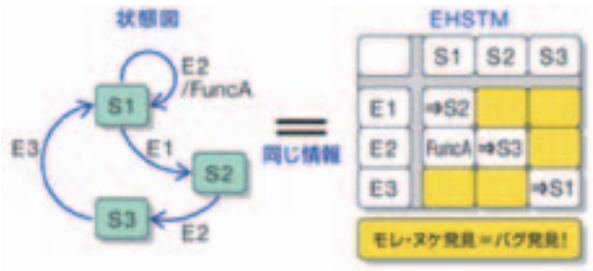
2. モデル検査ツールGarakabu2

我々は、ZIPCで記述された状態遷移表を振舞いモデルと捉え、その振舞いモデルのモデル検査ツールの研究開発をしています。名前はGarakabu2と言います。Garakabu2ではモデル検査で一般的な不変式検査、時相論理の他に、到達してはいけない不可セルへの到達検査、階層化された状態遷移表の検査にも対応しています。また、他のモデル検査ツールのような自然言語記述ではなく、ZIPCプロジェクトファイル（状態遷移表群）をそのままモデル検査出来るため、高ユーザビリティなモデル検査ツールといえます。また複数の状態遷移表のモデル検査を出来るためソースコードのようなシーケンシャルな状態遷移だけでなく並列な状態遷移のモデル検査を行うことが出来ます。ただし、幾つかの制限があります。ZIPCは色々なシミュレータ機能があり、またCASEツールとしてプログラムコード生成機能もあります。そういった機能に対する拡張記述に関しては対応していません。

［図4］　Garakabu2

3. 状態遷移表(ZIPC)に書けるコト

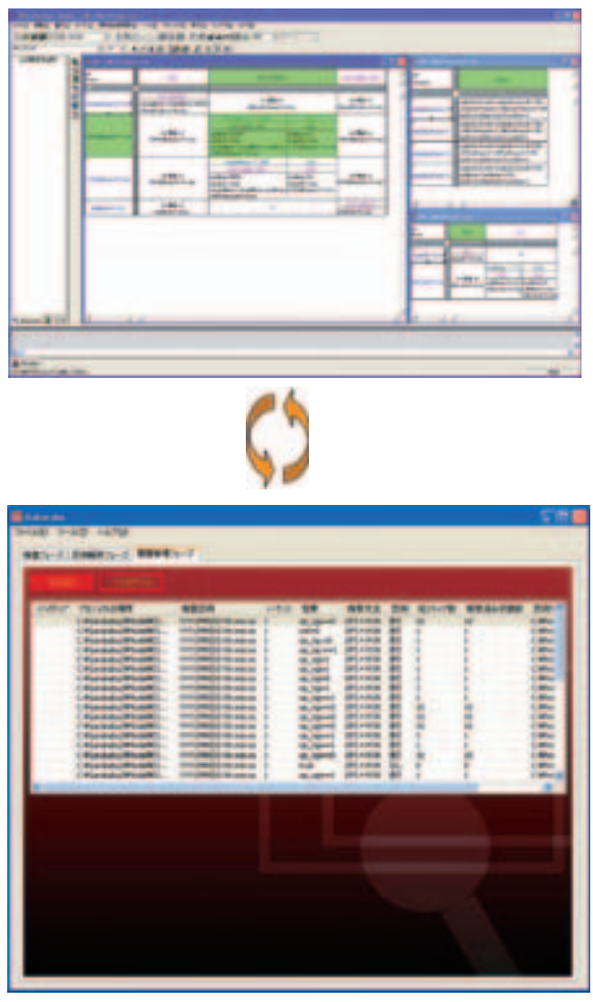
ZIPCは組込み向けCASEツールです。各タスク内にSTMの構成を記述するところまでを基本設計としています。我々はこのSTM内に粒度の大きな振舞いを記述して形式検証を行うことを目的としています。では、状態遷移表に記述出来ることとは何でしょうか? 左記にも述べましたが、状態遷移表には“振舞い”が記述できるのです。大きめの粒度での振舞いの検査をすることで、詳細設計化しプログラム・コード化する前の“考え方”や“安全性に対する取り組み”の正しさの検証が可能になります。ZIPCであれば、その後コード生成、シミュレーションの下流設計まで一貫した開発が行えます。MS-Excel等のオフィス・スイートでも状態遷移表の記述はできますが、モデルを記述して検証しただけで開発工程の連続性が確保できません。上記の理由でZIPCを、振舞いが“見える化”した状態で記述できるベスト・チョイスと考えました。また、状態遷移表を選択した理由がもう一つあります。状態遷移(振舞い)を記述する手法として、他に代表的なものに状態遷移図があります。UMLのステート・マシン・チャートなども状態遷移図の一つとして考えてよいでしょう。しかし、人は状態遷移図には“考えうる”遷移しか記述しないのです。考えていない状態と事象の組合せが漏れてしまうのです。



[図5] 状態遷移図と状態遷移表

4. 状態遷移表をモデル検査する“嬉しさ”

状態遷移表をそのままモデル検査すると何が嬉しいのでしょう? それには大きな3つの理由があります。まず、一つめは“見える化”した要求仕様に対する検証が出来る事。二つめは状態遷移表を書くことが振舞いモデル記述になっているので、形式検証(モデル検査)用の記述を行う必要がないこと。多くのモデル検査ツールは自然言語的記述で行われます。自然言語的な要求仕様書は、決して実際の開発現場では受け入れやすい方法ではないと考えます。三つめは、問題(モデル検査では反例と呼びます)が見つかったとき、その反例までの状態遷移の経路が見える化されているため、“どういう状態の流れで問題が発生したか”が非常に分かりやすいのです。反例が発生したときの設計へのフィードバックも容易です。この反例が起きたことが容易に分かることは非常に重要です。我々は状態遷移表からモデル検査ツールSpinへのコンバータの研究開発も行っていましたが、開発を中断しました。理由は、状態遷移表をSpinで読める形式(Promela言語)に変換して検査して反例を見つけても、その反例に対する状態遷移表へのフィードバックが予想以上に大変であり実用的ではないと判断したからです。



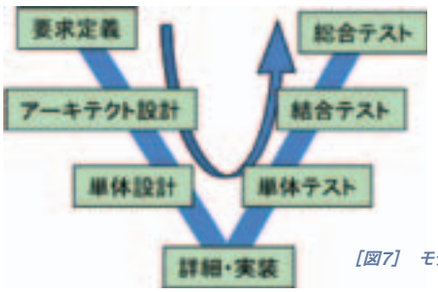
[図6] ZIPCとGarakabu2は連携している

5. MBD(モデルベース開発)との親和性

モデルベース開発については、色々なツールや取り組みが行われています。

ソースコードレベルで問題を発見し、そこで解決する。それ自体は製品として存在するソフトウェアの安全性や信頼性が確保できていれば問題ないのですが、次の設計に活かすことができない。新しいプロダクトや派生プロダクトで既存の資産を活かすことが難しく、設計と製造という作業の間に“大きくて深い川“がある。”ソースコード・デバッグでなんとかする“は、ソフトウェアの世界では“ものづくり日本”の悪しき遺産かもしれません。要求仕様から形式化されたモデル設計を行う、そして、それを要求仕様レベルで形式検証(モデル検査)し、正しい“振舞い“、”考え方“を起点に設計を進める。この出発点を確立することがモデルベース開発やV字開発と呼ばれる開発手法を成功させるポイントだと考えています。我々はこういった一貫したソフトウェア開発においても要求仕様の形式検証が貢献できると考えています。

“要求仕様から形式検証する“モデルベース開発においても、機能安全という意味においても、非常に重要なキーワードだと考えています。

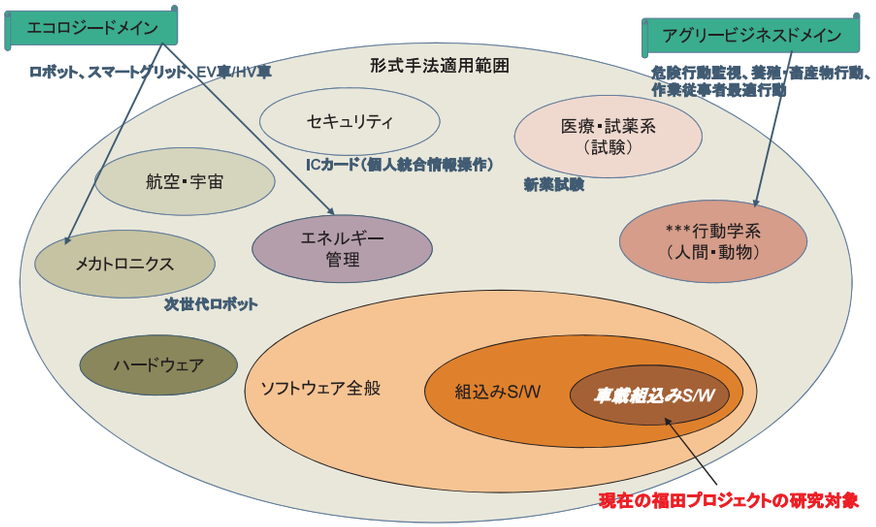


［図7］　モデルベースの一貫した開発

6. 将来、何が検証できるのか？

現在、形式検証(モデル検査)自体が組込みソフトウェアに限らずソフトウェア全般においても普及しているとは言えません。その要因の一つとして、モデル検査は全状態検査を行うため状態が大きいと“状態爆発”という現象が起きてしまい検査が中断してしまうことがあります。しかし、先に挙げた問題は、物理的な状態空間(具体的にはPCのメモリ容量等)の制約により発生しますが、今後、ハードウェアの進化、また、GoogleのMapReduceに代表されるようなクラスタリング・システムによって大規模な記憶領域(状態空間)や演算能力が手軽に使用できるようになったとき、状態遷移表による形式検証(モデル検査)は、機能安全を確保するためのごく当たり前な検証方法になっている可能性があります。

比較対象にはならないかもしれませんが、3次元CGの世界を例えると、かつて“旗のゆらぎの流体計算”を並列化された大型コンピューターを何台も使って行っていた処理よりはるかに高性能な3Dゲームが動くスマートフォンは、今、あなたのポケットの中にあります。形式検証を使用した機能安全は、意外とすぐ近くまで来ているのかもしれません。また、状態遷移表で振舞いモデルが記述できるということは、記述方法の精査は必要ですが、記述方式を拡張することによりソフトウェアに限らず、色々な“振舞い”の設計、検証が出来る可能性を秘めています。このことにより安全性・高信頼性を確保するための一貫としての機能安全を確保するという形式検証手法は多岐なドメインでの活用が考えられます。



［図8］　適用可能分野想像図

7. 課題

しかし、大規模なシステムの普及化が進むまで標準的な検証手法としての形式検証の実用化を待っているわけにはいきません。我々は状態爆発を起こしづらい手法の一つとしてSMT(Satisfiability Modulo Theories)技術を搭載しました。また、モデル検査エンジンの大規模クラスタリング・システムへの移植も検討しています。

8. 謝辞

最後に、今後、我々はより実用化された形式検証ツールの研究開発を邁進してまいります。形式検証という手法としてより開発現場でより実用的なものにするために、ご協力いただいているキャッツ株式会社の皆様、九州大学関係者、共同研究・ツール評価をしていただいている関係者様に感謝いたします。