タイトル	組込みシステムの試作における挙動の代数的記述と動作検証
著者	菊地，慶仁
引用	北海学園大学工学部研究報告，31：191-198
発行日	2004-02-20

組込みシステムの試作における挙動の代数的記述と動作検証

菊　地　慶　仁*

Algebraic Behavior Description and Validation in Early Design Phase for Embedded System

Yoshihito KIKUCHI*

要　旨

本研究では，設計初期段階におけるシステムの入出力関係や状態遷移規則を形式的に記述し，挙動の安定性や操作性などを分析することを目的とする。このために，代数的な仕様の記述と公理系に基づいた検証方式の提案を行う。また有効性の確認として，提案した方法に基づいて，状態遷移機械を対象とした状態遷移則の基本的な性質の検証を行う。さらに状態遷移が階層化されている場合について，記述方法と検証項目の提案及び仕様の検証を行う。

1．本研究の目的

情報機器開発の初期段階では，筐体や操作システムの意匠的なデザインに重点が置かれるが，今日では，開発期間の短縮化への要求から RapidPlus[1] などのヒューマン I/F の試作ツールを用いたデモンストレーションは行われている。しかし，開発初期段階で機器の設計仕様を表現する技術は確立されておらず，また製品仕様書も Word で書かれた文書がそのまま用いられていることが多い。このように，設計初期段階の仕様記述からの，定性的な検証と下流の開発工程への連携とが行われていない点が問題となっている。

本研究では，設計初期段階におけるシステムの入出力関係や状態遷移規則を形式的に記述し，挙動の安定性や操作性などを分析することを目的とする。このために，代数的な仕様記述と動作の検証方式の提案を行う。状態遷移則の基本的な性質の検証を行い有効性の確認を行う。さ

*北海学園大学工学部電子情報工学科
*Department of Electronics and Information Engineering, Faculty of Engineering, Hokkai-Gakuen University

らに状態遷移が階層化されている場合について記述方法と検証項目の提案を行う。

2．関連研究と本研究における課題

2.1 製品仕様の表現方式に関する考察

製品仕様の表現と検証のためには，様々な方式がある。デモンストレーションとシステムの振る舞い確認のために，最近ではRapidPLUS [1]のようなプロトタイピングツールも用いられており，幾つかのプロトタイプツールは，データをStatecharts [2] として出力できる。またGUI設計の仕様としてStatechartsを用いる幾つかの報告もある。しかしStatechartsは基本的には，図式表記法であり形式的な記述方式は確立されていないため，仕様の検証や下流のF/W開発へと繋げることは難しい。

また3次元CADシステムの形状データをViewPoint的なWeb 3 D技術によってプレゼンテーション用に再利用することも考えられる。しかしら，このアプローチは，設計初期段階でも形状データが用意されていることを前提としている。形状データが利用可能な場合でもアニメーションに関する情報は別途付与しなければならない。また，設計仕様の論理的な検証や後続するF/W開発への連携も実現されてはいない。

2.2 本研究での課題

関連研究から本研究の課題を以下にまとめる。

a）特定の形式に依存しない仕様記述能力

Statechartsは，有限状態機械を表現するためには有効であり[2]，製品仕様の記述に用いる報告もある[3]。しかしながら図式表記法であり，幾つかのリポート[5][6]は報告されているが形式的な表現方法は確立されていない。また状態同士の排他的な関係や，状態遷移が同期している状況を制約として表現する能力は用意されていない。従って，Statechartsを包含できるような汎用的な有限状態機械の記述方式が必要と考えられる。

b）仕様記述内容の非曖昧性

本研究で提案する方法論では，形式的な仕様の記述と論理的な検証を実現する必要がある。これらの目的のためには，数学的な明晰さと安定性が必要となる。従って最も望ましい表現方式は数式表現であり，また他の図式表記か言語的な表記方式から変換可能である必要がある。

c)下流開発工程との連携

開発スケジュールを短縮化し開発コストを低下するために，設計初期段階の製品仕様データを再利用することは非常に重要となる。この点からも仕様の記述は図式表記ではなく，他の形式に容易に変換可能であることが望ましい。さらに最終的な機能テストためのテストデータの生成に利用できることも重要と考えられる。

3．検証方式の提案

3.1 本報告における検証方式

本報告で用いる方法論は Hoare[7]が計算機言語 Pascal のために提唱した，公理によって表明を導出し，様々な目的の検証を行う方式である。Margren[8]は，図形描画プログラムによる描画結果が等価であるかどうかの判定を行っている。検証は次の 2 段階で行う。検証手順は IDEF 0 形式[9]で図 1 に示す。

1）入出力仕様の代数仕様記述の各宣言文一つ一つに，予め用意してある公理を適用して表明(Assertion)を生成する。全ての宣言文に適用すると，得られた結果が最終的な表明となる。ここでの表明は，(2)の検証に必要な情報を集積した幾つかのリストである。

2）表明に検証用の公理を適用して検証結果を得る。検証用の公理は，特定の目的毎に用意され表明を入力として検証結果を出力する。

検証結果は，真偽の判定値や特定の条件を満たす要素のリストなどである。判定の項目としては次のような情報が含まれる。

・有限状態機械として備えるべき性質（完全性，強連結性，状態の最小性）を満たしているか
・階層構造や並行状態記述の矛盾性の判定。

図 2 に，代数仕様，公理，生成される表明の関係の例を示す。代数仕様では，仕様記述の開始の宣言 Start と，有限状態機械で用いるアクション名を一つ宣言している。公理と表明は，2 つの宣言に関係する部分だけを示している。図の網掛けは，始めの宣言に対して，公理 1 を適用し，表明を生成した状態で，空のリストを生成している。次に 2 番目の宣言に対して公理 2

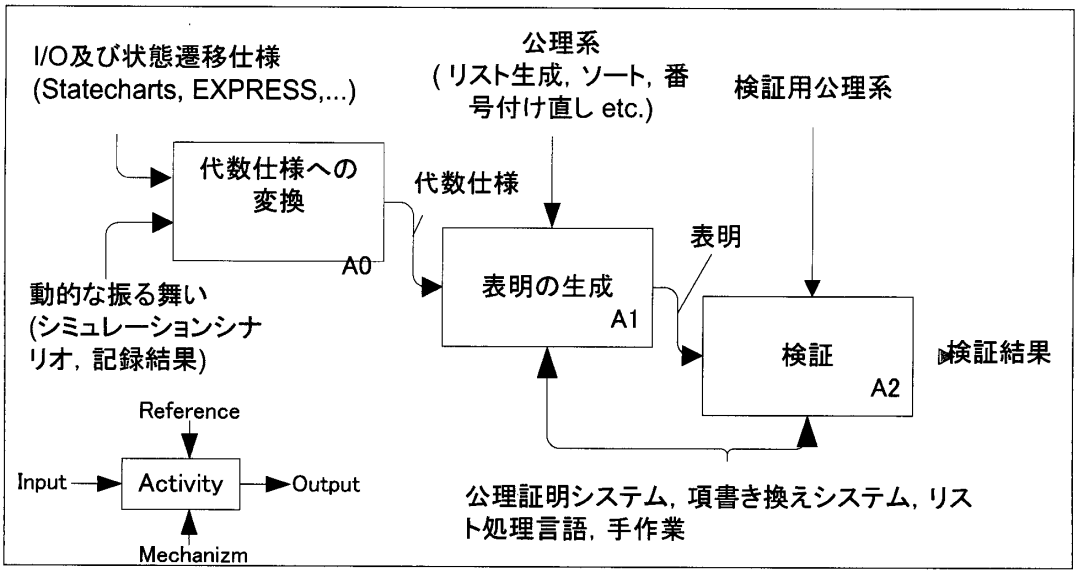


図 1　本研究で用いる手法の全体構成

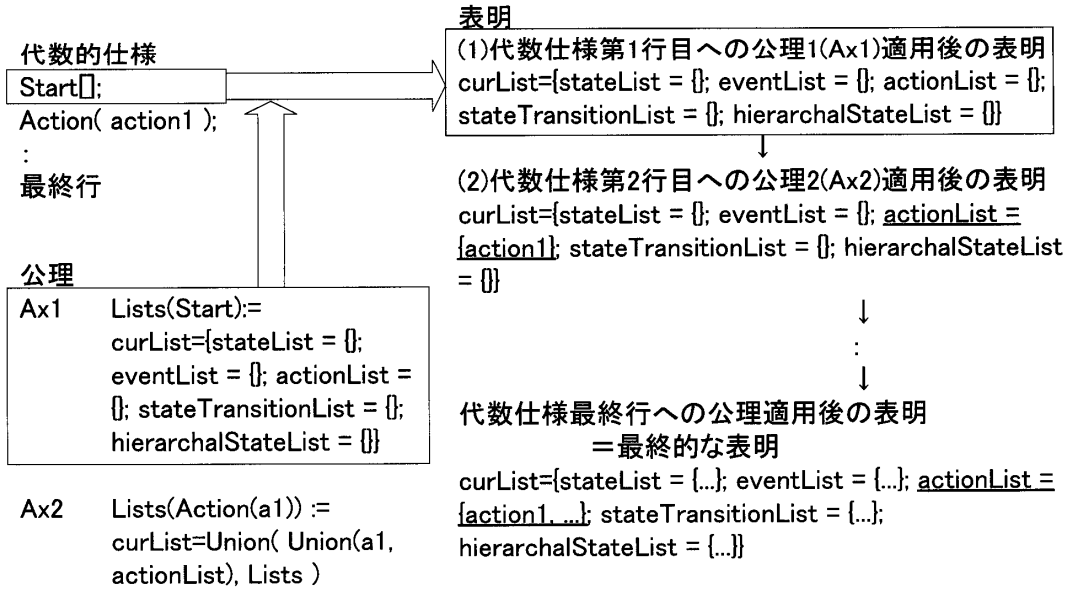


図2 代数仕様への公理の適用と表明の変化

を適用する。この公理は，空のリストの一つに Action 1 という名前を一つ加え，更に全体のリストに加える。表明中の下線部分のリストに新しい要素が加えられている。

このように，宣言の一つ一つに，対応する公理が適用され，全ての宣言への公理の適用が終わると，残された表明が最終的な仕様の表現となる。この表明に検証用の公理を適用して，仕様の検証を行う。

3.2 本方式の特徴と利点

本研究の方法論には，次の特徴と利点がある。

1）この方法論は，数学的な明晰さを持ち，特定の記述方式には依存しないため他の記述方式に変換することが容易なので再利用性が高くなる。また計算機による実装が行われていない場合でも数学的に解くことができる。

2）システムの静的な仕様と，動的な振る舞いを同じ方式で検証することができる。ただし，静的／動的なモデルでは異なった公理系を用意する必要がある。

3）この方法論に基づくことで，有限状態機械モデルのための規格を定義することができる。一般にはシステムの振る舞い全体が規格通りかを判断することは難しいが，システムが公理で記述された一つ一つ動作を満たしているかどうか判定することは容易である。このことによって，システムの振る舞いが規格通りであることを保証できることになる。

このような特色を持つ検証方式に関する報告はあまり行われておらず，本研究の独自性を示していると考えられる。

4．状態遷移の検証

4.1 記述内容

システムの状態遷移は

・状態名，イベント名，アクション名の定義

・状態の遷移則

・状態の階層構造

などで定義される。階層構造の表現は，次の項目に関して行っている。

・スーパー状態と，その中に含まれるサブ状態

・初期状態，最終状態，ヒストリ・インディケータなどの情報

また，次節の基本的な性質の検証を行うために，全ての宣言に対して公理が適用された後で，状態推移の可能性を全て列挙したリストも生成する。

4.2 状態遷移の基本的な性質の検証

生成された表明に対して次のような基本的な性質検証を行っている。

1）完全定義

任意の状態からの遷移先の状態と出力が最低一つ定まっているかどうか

2）強連結性

任意の2状態u，vに対して，uからvに到達でき，かつvからuに到達できるかどうか

3）状態数の最小性

あるイベントとアクションの組み合わせで遷移が発生する状態の数が最小かどうか。同じイベントとアクションの組み合わせで複数の状態からの遷移が複数存在する場合，本来は単一であるべき状態を複数宣言してしまっている可能性がある。

4）状態遷移の決定性

ある状態からガードまでも含めた同一イベントで別々の状態への遷移が定義されている場合，この定義は非決定性を持つことになり，仕様としては曖昧性を持つことになる。

4.3 階層化された状態遷移の検証項目

図3に階層化された状態遷移の例を示す。この図では，s4がスーパー状態であり，s5とs6をサブ状態として含んでいる。ふち無しの黒丸は，s4への状態遷移が宣言されている場合に，実際にはs5に遷移することを示す初期状態を宣言している。ふち付きの黒丸への遷移は，全てs4から外部の特定の状態への遷移として扱われることを示している。丸にHが入っている記号は，以前にs4内部での状態を記憶して次に丸Hへの遷移が発生した場合には同じ状態から開始されることを示す，ヒストリ・インディケータを宣言している。このような記述に対して，次のような記述内容の検証を行っている。

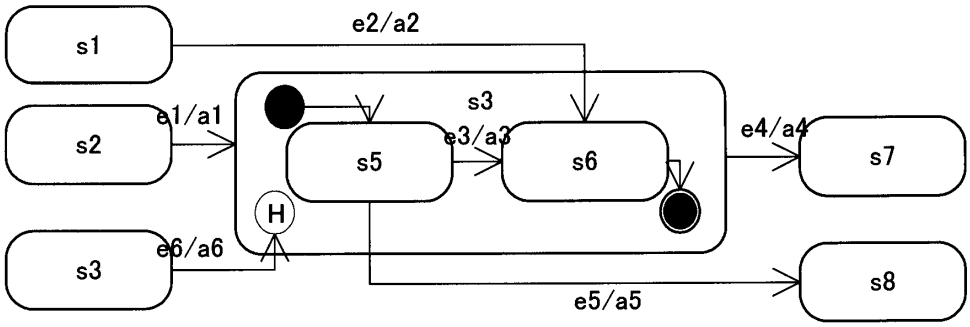
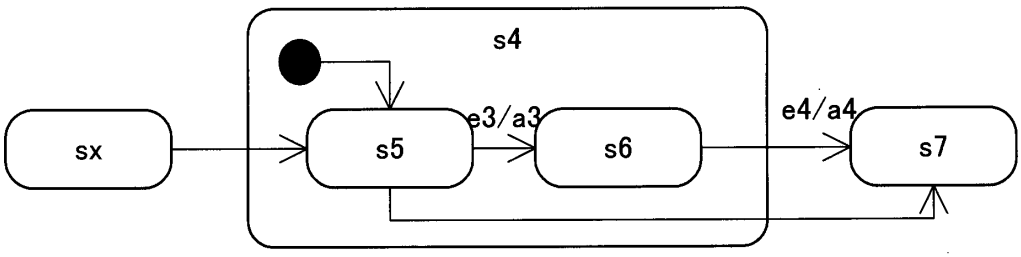


図3　状態遷移の階層構造の例



初期状態が宣言されているのに，s5 への遷移を直接記述すると，初期状態がs5から別の状態に変わった場合に対応できない．

外部への遷移がすべて同じ状態に対してならば，最終状態の使用を検討すべき．

図4　間違いではないが一貫性に問題がある記述例

1）記述に矛盾が無いかどうかの検証

・スーパー状態／サブ状態のネスティングが矛盾していないか(図の例では，スーパー状態ｓ4とサブ状態ｓ5もしくはｓ6で同一の状態名が用いられるか，ｓ5，ｓ6のサブ状態名が別の階層でもサブ状態名として使われている場合，矛盾があることになる)

・サブ状態から外部への遷移が最低一つ定義されているかどうか

・スーパー状態へ遷移がある場合に，サブ状態の一つが初期状態として宣言されているか（図の例では，ｓ2からｓ4への遷移が宣言されているが，この場合にはｓ5又はｓ6のどちらからが初期状態として宣言されていなくてはならない)

・最終状態が使われている場合に，外部の状態の一つが遷移先として定義されているか

2）モデルの構造の一貫性に関する検証

このグループでは，完全に矛盾しているかどうかではなく，記述内容が一貫しているかどうかを検証している。下の検証内容の一部を図4に示している。

・開始状態，終了状態として宣言された状態に，開始／終了状態を用いないで直接遷移しているかどうか。このように開始／終了状態を用いないで遷移を記述していると，開始／終了状態の対象が変更された場合に，同一状態への遷移とならない可能性がある(図4左側)。

・スーパー状態に遷移した時に，全ての遷移が同じサブ状態から開始している場合には，初期状態を使用することの推奨。

・同じく，外部の状態への遷移が全て同じ状態に遷移する場合には，最終状態を使用することの推奨(図4右側)。

・階層からの抜け方の一貫性の検証。

また上記の検証は階層構造が入れ子になった状態でも判定できる必要がある。また並行動作状態を検証するためには，andサブ状態の記述のために自動的なラベル付けが必要になる。

4.4 階層構造の表現と検証

ここでは，図5以下にVWCの状態遷移仕様の検証を行った例を示す。

1）この仕様では，まず「電源ON」中にサブ状態「待機中」があるが，※印の状態「UNITcurrに接続中」「自UNITに接続」の中でも同じ状態名が別個の目的で使われている。本来は，同じ状態名があるスーパー状態に含まれる場合に，他のスーパー状態名の下に含まれることは間違いとなる。

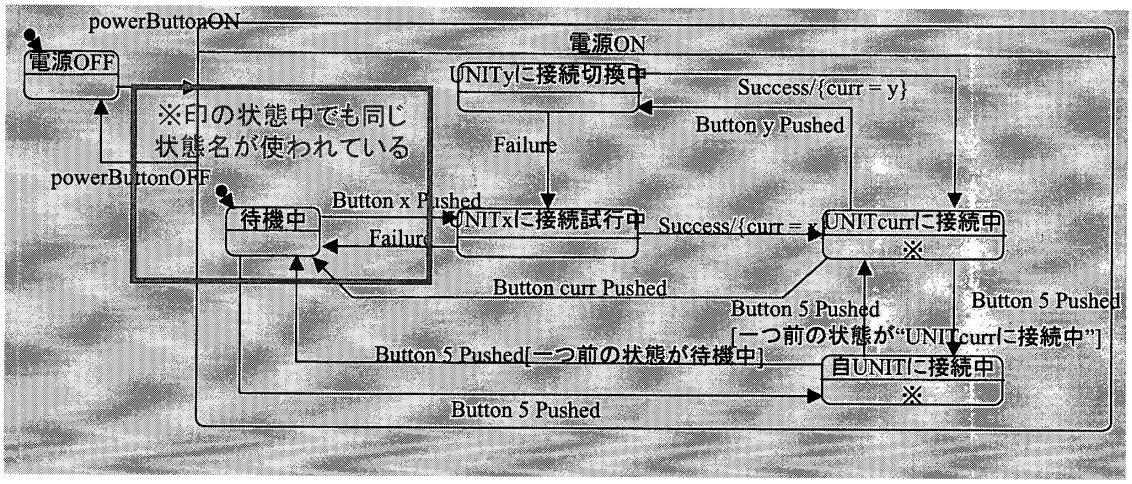


図5　VWC状態遷移仕様（枠中の状態名「待機中」が※印の状態「UNITcurrに接続中」「自UNIに接続」の中でも使われている）

```
HierarchalState|{"電源On",
  {"待機中", "Unitxに接続切換中", "Unitxに接続試行中", "UnitCurrに接続中",
   "自Unitに接続中"}, {"use_start_state", "待機中"}
  }|;
HierarchalState|
  {"UnitCurrに接続中", {"待機中", "映像設定1", "映像設定2", "映像設定3"}}|;
HierarchalState|
  {"自Unitに接続中", {"待機中", "映像設定1", "映像設定2", "映像設定3"}}|;
```

図6　上記状態遷移仕様の代数的表現（枠中に同じ状態名が出現している）

2）この仕様の代数表現を図6に，その検証結果の例を図7に示す。仕様と検証結果は非常に長いものみなるので，ここでは一部のみを示している。

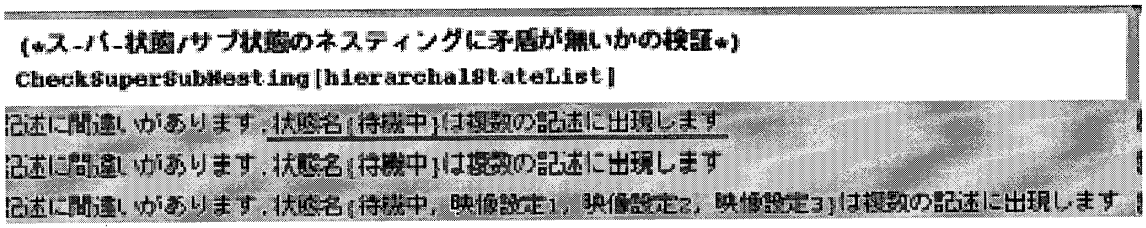


図7　上記代数記述の検証結果

5．結　　論

本研究の成果は，以下にまとめられる。

1）設計初期段階における情報機器の挙動の代数的な記述と検証を行う方法論を提案した。

2）状態遷移の記述に対して基本的な検証を行えることを示した。

3）階層化された状態遷移を表現する方式を提案し，検証項目を示し，実際に検証を行った。

今後は次のような問題を解決する必要がある。

1）or サブ状態を拡張した and サブ状態の表現，状態の同期や排他的関係などの記述と検証方式の確立

2）静的な仕様の検証だけでなく，実際にシステムが動的に動作する場合の検証方式の確立

参 考 文 献

1．http://www.e-sim.com/home/index-h.htm

2．David Harel, Statecharts: A visual formalism for complex systems, Science of Computer Programming 8 (1987) 231-274 North-Holland.

3．H.E.エリクソン他，UMLガイドブック，エス・アイ・ビーアクセス，2000年1月

4．Ian Horrocks, Constructing the User Interface with Statecharts, Addison-Wesley, 1999.

5．A Pattern Language of Statecharts, Sherif M. Yacoub; Hany H. Ammar, PLoP-98 Conference.

6．Implementing Statecharts using Extended Positional Grammars, G. Costagliola, V. Deufemia, F. Ferrucci and C. Gravino, VLFM01, Symposium on Visual Languages and Formal Method

7．C. A. R. Hoare, N Wirth, An axiomatic definition of the programming language Pascal. Acta Information 2, 1973, 335-355

8．William R. Margren, Formal Specification of Graphic Data Types, ACM Transactions on Programming Languages and systems, vol. 4, No. 4, October 1982, 687-710.

9．Draft Federal Information Processing Standards Publication 183, Announcing the Standard for INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF 0) 5, 1993.