Vol.2010-SE-170 No.7
2010/11/11

ドメイン知識を用いた検証に向けた状態遷移図の抽象化方法に関する考察

園田貴大[†1]　大森洋一[†1]
日下部茂[†1]　荒木啓二郎[†1]

組み込みソフトウェアは日常の多くの範囲で利用されるようになり，それに伴った開発規模の増大，高い信頼性の要求が生じている．このような状況に対して様々な取り組みがなされている．形式手法に含まれるモデル検査もその一つである．モデル検査は検証対象の状態に対して網羅的な探索を行うが，実用規模のシステムに対してモデル検査を行うと，状態が多すぎるために検証を終えることができないことがある．そこで本論文では，抽象化を行う際に検証対象に関する知識を利用した抽象化の手法を提案する．この提案手法を用いることで，従来の手法を適用する事が困難な場合でも検証目的の性質を維持した抽象化ができるようになる．

Examination about abstraction for state transtion diagram to velification using domain knowledge

Takahiro Sonoda,[†1] Yoichi Oomori,[†1]
Shigeru Kusakabe[†1] and Keijiro Araki[†1]

Recently embeded software are using large fields in daily life. So developing scale is increasing and high reliability is needed. In these situation , many approaches being performed, and model checking included formal method is one of this. Model checkin explore in target system state exhaustively. But if target system is practical using system , it have too large state to explore so model checkin can't finish. To solve this problem , we suggest abstract method using knowledge of tartget system . Using this method in situation where hard to apply traditional abstrac method , you can get abstracted model keeping property to velification.

1. 最　初　に

近年の組み込みソフトウェアは自動車やエレベーター，携帯電話など広い範囲に使われ，より日常的なものになっている．利用範囲の拡大に従い，組み込みソフトウェアに求められる機能や規模も増大しており，開発にかかる工数やコストが増加している．一方で，組み込みソフトウェアで発生した障害が原因で人身に関わる事故が起きているケースや，製品を販売した後で生じた障害への対応のために莫大な費用がかかった事例も存在する．組み込みソフトウェアの品質に対する信頼性が社会的な問題となっている[1)]．

このような状況に対して，従来の経験に従った開発手法では対応しきれない状況が生じている[2)]．この問題を解決するために様々な取り組みがなされており，組み込みソフトウェアの分野ではプロダクトライン開発の研究や，国際規格（自動車分野における ISO26262 など）の制定などが行われている．形式手法についての取り組みも同様であり，形式手法を導入して効果を挙げている事例も存在する[3)]．形式手法には，形式仕様記述と形式検証がある．形式仕様記述とは，数学に基づいた形式仕様記述言語を用いて仕様の記述を行うことで，厳密な仕様の記述を目的とする．形式検証とは，ソフトウェアの振る舞いの検証を目的としている．数式や論理式でシステムのモデルを記述し，ソフトウェアに障害が無いことを検証する手法である．振る舞いのモデルを手法に応じた記述用の言語で作成し，そのモデルが取りうる状態を網羅的に探索する手法をモデル検査と言う．モデル検査を行うことで，開発するソフトウェアの信頼性を高めることが期待されているが，モデルの抽象化や満たすべき性質の記述方法などに課題が残っている．

2. モデル検査

この章では，まずモデル検査の概要を述べる．次に，モデル検査を行う上で用いられる一般的な抽象化の手法と，抽象化手法における現状の課題について述べる．

2.1 モデル検査

モデル検査は，集合論や記号論理などの数学的記述で仕様を表現する形式手法の一つである[4)]．モデル検査では，対象のシステムが持つ変数の組み合わせを用いて，状態遷移を網羅的に探索する．この手法を用いることで，ある状態に到達する可能性（到達可能性）や到達

†1 九州大学
Kyushu University

しない可能性（到達不可能性），状態の遷移が着実に行われてデッドロックに陥らないことを考慮する進行性などを検証する事が出来る．この様な探索を行うには，モデル検査の中で用いる手法で規定された言語でモデルを記述する必要がある．例えばモデル検査手法の一つである SPIN には Promela という記述言語が対応している．

一般的なモデル検査の流れは次の通りである．

(1) 用いる手法で規定された言語でモデルを作成する
(2) 検証したい性質を表す式を LTL などの時相論理式で記述する
(3) モデルと式を用いて検証を行う
(4) 検証を行った結果，
  (a) 仕様を満たしていると判断すれば終了とする
  (b) 仕様を満たしていないと判断した場合，反例として検出された結果を検討し，モデルや式の修正を行って再度検証を行う

しかしながら，仕様や設計段階のシステムをそのままモデル検査用の言語で記述したのでは検証が不可能になる場合がある．これはモデル検査が状態を網羅的に探索する際に，状態の数が多くなり過ぎることによる状態爆発が起こるためである．この問題を解決するためには，検証対象のシステムを適切に抽象化する必要がある．

2.2 抽象化の手法

この節では，システムの抽象化行う際に用いられている一般的な抽象化の手法について述べる．

データマッピング

データマッピング[5] とは，システムを表すのに利用する各変数に対する抽象化の手法であり，データ抽象とも言われる．適応対象の変数が取りうる値を範囲分けし，それぞれの範囲の値を適当な一つの値へと写像するものである．良く用いられる例として，整数全体を表す変数に対するデータマッピングの例を挙げる．整数全体を表す変数 x と，ある値 i の大小のみを考慮したい場合，以下の様な方法で抽象化を行う．

$$Mapping : M(x) = \begin{cases} Large & (x > i) \\ Equal & (x = i) \\ Small & (x < i) \end{cases}$$

i が 0 のとき，このマッピング関数 M は x の符号を判定していることになる．このようにデータマッピングによって，広範囲な状態を持つ変数を範囲分けして少数の値で表すことで状態を削減する事ができる．

述語抽象

データマッピングは単一の変数のみに対して行われる手法であり，複数の変数によって構成される述語 (遷移のガード条件など) に対しては適応する事ができない．そこで，各述語が満たされるか否かのみを判定することで抽象化を行うのが述語抽象[6] である．すなわち，ある条件を表す述語 ϕ i に対して，

$$\epsilon i = \begin{cases} T(\phi i \; is \; satisfied) \\ F(\phi i \; is \; not \; satisfied) \end{cases}$$

とするものである．例えば変数 a,b を用いて

$v : (a > 5 \wedge b < 1)$

という条件を表す場合，

$$\epsilon 1 = \begin{cases} T(a > 5) \\ F(a \leq 5) \end{cases}$$

$$\epsilon 1 = \begin{cases} T(b < 1) \\ F(b \geq 1) \end{cases}$$

とすると，

$v : (\epsilon 1 = T \wedge \epsilon 2 = F)$

と表すことが可能となる．述語抽象を行う際には，対象となる述語の選択が重要である．述語の選択と述語抽象を用いたモデル検査を繰り返し, 反例を分析することで抽象化の精度を向上させる CEGAR(CounterExample-Guided Abstraction Refinement) も存在する．

スライシング

モデル検査におけるスライシング[7] とは，検証したい性質に着目し，特定の状態とその状態と依存関係にある状態を全体から抽出することで抽象化を行う手法である．特定の状態に対し，その状態に影響を与える状態の集合をバックワードスライス，状態から影響を与えられる状態の集合をフォワードスライスと呼ぶ．一般にはフォワードスライスに比べてバックワードスライスの方が利用されることが多いため，バックワードスライスをスライスと呼ぶこともある．

2.3 現状の課題

抽象化行う様々な手法があるものの，それらの手法を適用できない，または適用するのが難しい場合がある．データマッピングと述語抽象については，抽象化対象の変数の取りうる値が少ない場合は抽象化の効果が薄い．例えば Bool 値などである．この様な時には，その場その場のやり方で抽象化が行われているのが現状である．抽象化を行う際には，着目する性質を決めた後でその性質に影響の無い状態空間や，実現する内容の詳細を捨象して状態数を削減する．このような抽象化の方法においては検査対象システムの振る舞いに関する知識が重要であり，一般的な議論が難しい[4]．抽象化の方法を間違えてしまうと抽象化後に検証したい性質が維持されず，正確な検証を行うことができない．検査対象のシステムに関する知識を利用した上で，検証する性質が抽象化後も維持されている抽象化の手法が求められている．

3. 提案する手法

本章では，状態遷移の過程である状態に達することが出来ないこと（到達不可能性）を安全性の一つと定義する．提案する手法では抽象化の最初にモデルが検証する性質を保持していることを保証する．抽象化の際には検証対象システムが行う動作に付いての知識を要する．

3.1 前 提 条 件

手法の提案に際し，適応対象は UML2.0 のステートマシン図に従ったものの中でも，更に記述内容を限定したものを対象とする．具体的な記述内容を以下に述べる．

状態の持つ要素は，

- 状態名：状態の固有名称
- 入場動作 (entry)：前状態から遷移した直後に実行される動作
- 実行活動 (do)：状態である間継続的に実行される動作
- 退場動作 (exit)：次状態に遷移する直前に実行される動作

である．

遷移の持つ要素は

- イベント：遷移の発火条件となる動作
- ガード：イベントが生じて遷移が行われる際に満たされているべき条件
- アクション：遷移が行われる際に行われる動作

である．

ここで状態の持つ要素のうち入場動作 (entry)，実行活動 (do)，退場動作（exit）の各動作を合わせて「状態の持つ動作」と定義する．また，遷移の持つ要素を合わせて遷移内容と定義する．

状態と遷移の具体的な表記を図 1 に示す．

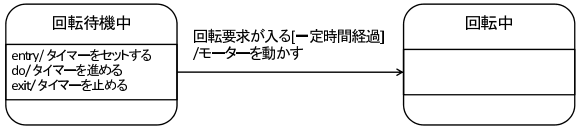

図 1 UML2.0 における状態と遷移の記述例

組み込みソフトウェアにおいては，実時間の概念や遷移の優先度を記述する場合が多数存在する．本論文ではそれらの遷移内容については対象外としている．

また，本論文では，特定状態への到達不可能性を安全性の一つとしている．ソフトウェアの設計の際に正常な動作とは異なる異常な状態を定義する．その状態へ到達することが無いことを保証できれば異常な状態にならないという安全性を保証できるからである．例えば，電動式ドアの組み込みソフトウェアに対して「物が挟まった状態ではドアが閉まらない」という安全性を考えるとする．この場合，ドアに物が挟まったまま閉まっている状態とその状態に至る遷移をを考る．その状態に到達する事が不可能であることを検証すれば良い．

3.2 抽象化の方向性の決定

状態遷移の到達不可能性を検証するためには，

- どの状態からどの状態への到達不可能性を検証したいのか？
- その過程でどの様な状態を辿る可能性があるのか？

を考える必要がある．そこで，抽象化の最初の作業として，対象システムの動作を列挙する．列挙する過程で，システムがどの様な動作を行うのかを考える必要がある．

次に，列挙する各動作に対しその動作が始まる状態と，その状態から遷移する可能性のある状態を作成する．

各動作に対する状態と遷移をしたら，それらの状態の遷移元と遷移先の関係から各状態を遷移で結び，抽象化した状態遷移図を作成する．この際，抽象化前のモデルを参考にすることで動作を列挙した際の漏れを防ぐことや，逆に余剰な動作を発見する事ができる．

この時に挙げる動作の抽象度によって，その後のモデルの抽象度や抽象化の方向性が大筋

決定する．ここで言う抽象化の方向性とは，どの動作や状態に着目して抽象化を行うか，ということである．

3.3 遷移内容と状態の持つ動作の統合

次に遷移内容と状態の持つ動作の抽象化を行う．抽象化の際には，抽象化前後の状態の対応付けを行う．抽象化前の状態が，先に作成した状態遷移図のどの状態に対応するのかを一つずつ考えていく．対応付けた後の遷移関係には幾つか種類があるため，それらを分類して抽象化の方法を考える．本論文では遷移内容を下記に記すように扱う．

(1) 抽象化後も他の状態との統合や複数の状態への分割が無い場合
そのままの遷移を扱う

(2) 抽象化後に他の状態と統合される場合
繊維関係には以下の 3 つの場合が考えられる
(a) 統合する状態が遷移先となっている場合
抽象化後の状態を遷移先として扱う
(b) 統合の際に遷移元となる状態が含まれる場合
抽象化後の状態を遷移元として扱う
(c) 統合する遷移間で状態遷移が行われている場合
遷移を削除する

(3) 抽象化後に複数の遷移へと分割される場合
(a) 遷移先となる状態を分割する場合
分割後の全状態を遷移先として遷移する
(b) 遷移元となる状態を分割する場合
分割後の全状態から，分割前の遷移先の状態へと遷移する

また，状態が持つ動作に対しては，下記の様に扱う

(1) 抽象化後に他の状態と統合される場合
抽象化後の状態に，所持している動作を持たせる
統合する他の状態が動作を所持する場合は，それらも含めて持たせる
この場合，抽象化後の状態は動作として複数の可能性を持つ

(2) 抽象化後に複数の状態へと分割される場合
抽象化後の全状態に，所持している動作を持たせる

上記の処理を各遷移内容と各状態の持つ動作に対して行っていくことで，状態遷移の抽象化を行う．提案する抽象化の手法では，検証する性質は到達不可能性としている．到達不可能性の検証のためには「抽象化後のモデルが状態に到達できなければ抽象化前のモデルも到達できない」ことが保証できていれば良い．そのため，到達することのできる状態を増やす方向で行う抽象化は妥当であると言える．検証の結果，状態に到達する事ができないことが分かれば，抽象化前のモデルもその状態に到達する事が無いことを保証できる．

一方で,「抽象化後のモデルが状態に到達できれば抽象化前のモデルも到達できる」ことを保証する性質は到達可能性と言う．この性質もソフトウェアの安全性を表す上で用いられることがあるが，提案する手法では対象外としている．

4. 提案手法の適用事例 -電動クローザシステム-

本章では，3 章で提案した手法を実際に利用されているシステムに対して適応した事例を紹介する．

4.1 対象システム

適応対象のシステムとして，自動車のトランクに付いている電動で鍵のオープンとクローズ行うクローザシステムを扱う．このシステムの動作を表す状態遷移図を図 2 に示す．ただしこの図では各遷移の内容と状態が持つ動作は見易さのために省略している．特徴としては，オープン動作とクローズ動作それぞれの途中で待機状態があり，各動作が幾つかの段階に分かれていることである．各動作の待機状態は，それぞれに続く動作を行うことができるのかを判定するための状態である．以降の節では，図 2 の状態遷移図に対して提案する手法を適用した結果を述べていく．

4.2 抽象化の方向性決定

最初に，今回の事例でシステムの動作を列挙する．検証するために挙げた動作は以下の５つである．

- オープン動作が始まったら，オープン動作が完了すること
- オープン動作が始まる際に，オープン動作の開始条件が満たされていること
- クローズ動作が始まったら，クローズ動作が完了すること
- クローズ動作が始まる際に，クローズ動作の開始条件が満たされていること
- 反転してオープンの状態になったら，オープン動作が始まること

これらに対応した状態及び状態遷移をそれぞれ考え，状態遷移図を作成する．

例として,「クローズ動作が始まったら，クローズ動作が完了すること」という動作を扱

Vol.2010-SE-170 No.7
2010/11/11

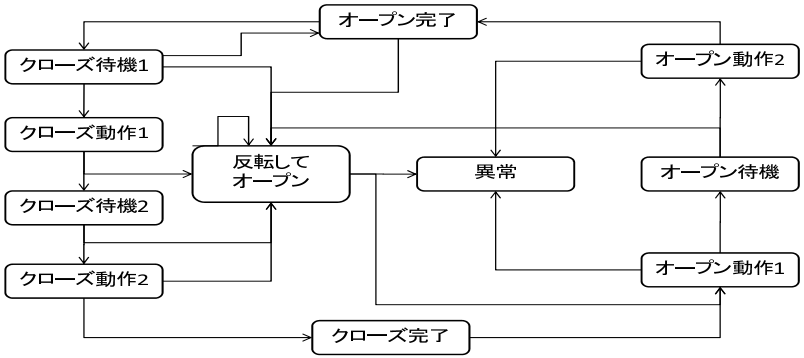

図 2 対象とする電動クローザシステム

う．この動作では，クローズ動作が始まっていることが前提であるため,「クローズ動作」の状態を持つ．この状態から遷移する可能性のあるを状態を考える．動作には成功した場合と失敗した場合がある．成功した場合は，クローズが完了するため,「クローズ完了」状態へと遷移する．動作が失敗した場合を考えると，失敗したタイミングから 2 通りの遷移が存在する．1 つは鍵が閉まる前半で失敗した場合であり，もう 1 つは鍵が閉まる後半で失敗した場合である．ここで，前半と後半の区別は，鍵が途中までかかって，いわゆる半ドア状態になる前が前半で，それ以降が後半である．そして，前半後半それぞれの状態で失敗した遷移先が「オープン完了」と「反転してオープン」の状態である．この例では，動作が完了した後の状態を考えるのは比較的容易だが，動作が失敗した場合を考えるのには対象システムについての知識が必要である．

結果として作成した状態と遷移の図を図 3 に示す．

列挙した全ての動作に対して図 3 のような状態遷移図を作成し，統合して一つの状態遷移図を作成する．今回の事例では，オープンとクローズの各動作については待機状態での判定を行っていない．これは列挙した動作の中に，判定に関する動作が挙げられていないためである．そのため，待機状態とその前後の動作状態をまとめて一つの状態として扱うことが出来た．このように，どこまで詳細な動作を考慮するかでモデルの抽象度を決定する事が出来る．

4.3 遷移と状態の持つ動作の抽象化

抽象化した状態と遷移を作成することで抽象化の方向性を決定した後で，遷移内容と状態

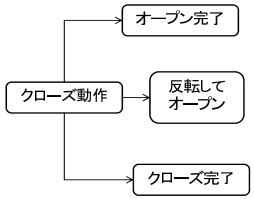

図 3 「クローズ動作が始まったらクローズ動作が完了する事」の状態と遷移

の持つ動作の抽象化を行う．本論文では，遷移内容としてイベント・ガード・アクションを扱うこととしているが，そのうちアクションは今回検証する上で遷移に影響を与えることがないと判断出来たので省略した．抽象化前後ので状態の対応付けを行った．結果を表 1 に示す．

表 1 クローズ動作が始まったらクローズ動作が完了する事」における抽象化前後の状態の対応付け

抽象化前	抽象化後
クローズ待機 1	クローズ動作
クローズ動作 1	
クローズ待機 2	
クローズ動作 2	
オープン完了	オープン完了
反転してオープン	反転してオープン
クローズ完了	クローズ完了

遷移内容と状態の持つ動作は，抽象化前の状態遷移図を基に抽象化を行う．抽象化前の遷移内容や状態の持つ動作を参考にしながら，3 章に示したやり方で抽象化を行った．結果を図 4 に示す．

図 4 に示した様な今回の抽象化方法では，抽象化前のモデルでは到達できない条件でも到達してしまう可能性がある．これは，抽象化前よりも遷移が増えたとみなせる状態が存在するからである．例えば抽象化前の「クローズ待機 1」は抽象化後に「反転してオープン」への遷移を持っているとみなすことができる．抽象化後のモデルが状態に到達した場合は，到達した経緯を分析した上で再度抽象化を行い検証する必要がある．

この章の最後に，図 2 を抽象化したものを図 5 に示す．この図では，各状態が持つ動作は

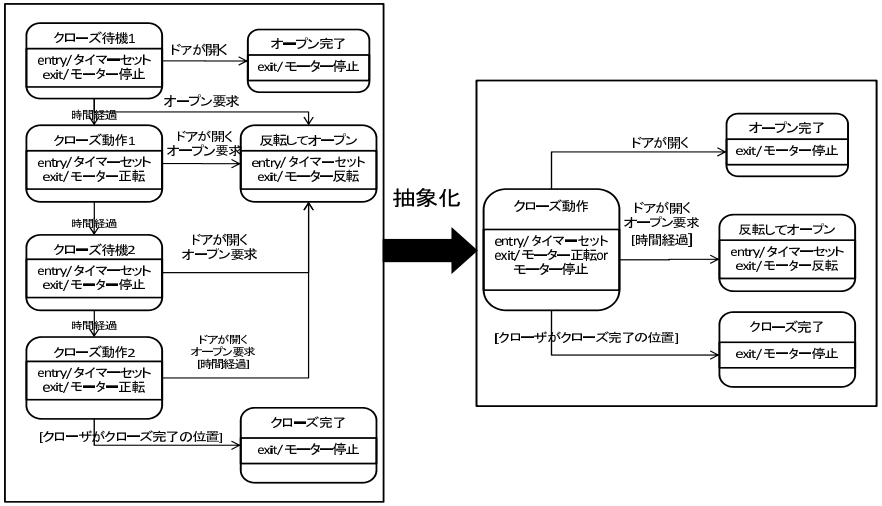

図 4 「クローズ動作が始まったらクローズ動作が完了する事」に対する遷移内容の抽象化

省略している．今回の状態遷移図では，各状態が持つ動作が状態間の遷移のイベント，ガードに対して影響を与えることが無かったためである．

5. 考　　察

本論文で提案した手法は，述語抽象やデータマッピングなどの抽象化手法が適用できない，または適用するのが難しい状況を想定している．その上で，抽象化の方針を与えるのが本論文で提案した手法の主目的である．そこで，提案した手法を事例に適用した結果を基に，提案した手法を用いる場合と用いずに適宜抽象化する場合を比較して考察することとする．

5.1 抽象化の方向性の決定について

列挙した動作を基に抽象化を行えることは提案手法の目指す一番の目的である．動作を列挙するためには対象システムに関する知識が必要であるが，逆に言えば，対象システムに関する知識があれば抽象化を行えるということである．

経験に従った従来のやり方では，対象システムの知識を用いる方法が分からない事があるかもしれない．例えば，今回の事例を経験に基づいた従来のやり方で抽象化することを考える．実際に，著者は今回の提案手法を考案する前に抽象化を行った．その際には

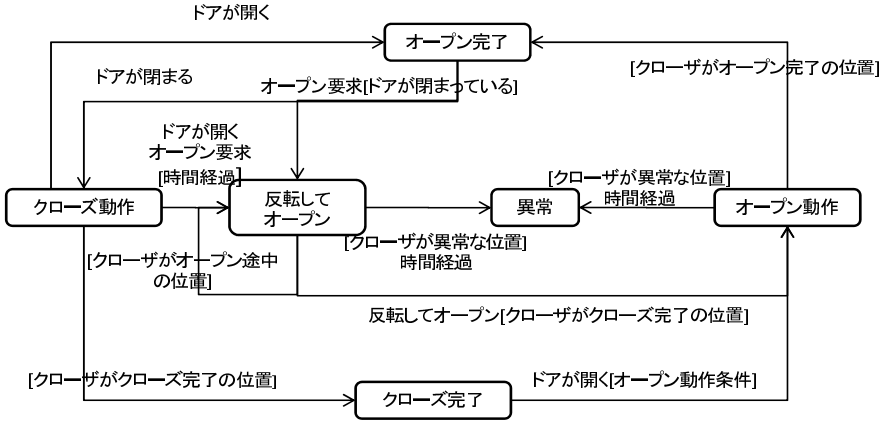

図 5 抽象化後の対象システム

- オープン，クローズ，異常の 3 つの状態で十分ではないのか
- クローズ動作を，クローズ待機 1 とクローズ動作 1，クローズ待機 2 とクローズ動作 2 の 2 つの状態として扱って良いのではないか

という考えがあった．そして，この抽象化の方向性に対し，明確な対案が無かったのである．しかしながら提案手法において動作を列挙していく上で，各動作が始まる前に開始条件が守られていることを確認することや，オープン動作が 1 つの状態として統合されているため，クローズ動作も同様に 1 つの状態として統合するという案が生じた．提案手法では抽象化前の状態遷移図を参考にしながらも，対象システムの動作自体を抽象的に捉えることができるという利点がある．

一方で，抽象度に関する課題が残る．列挙する各動作の抽象度や，その他の動作との釣り合いを持たせることが重要である．本論文で述べた事例の際には，オープンとクローズと言った対になる動作を考えることや，動作で異常が起きた際の事を考慮することで対応した．経験に従ったやり方なら無意識にされている可能性があるが，この様な観点をまとめていくことで，抽象化のバリエーションを増やす事が出来る．まとめた観点を参考にしていくことで，動作を列挙する際の見落としや挙げる動作の抽象度のばらつきに気付くことが出来るなどの利点を得ることができる．

5.2 遷移内容と状態の持つ動作の抽象化について

提案手法に従って抽象化を行うと，存在する全ての遷移内容と状態の動作を保存することになる．それどころか，状態が分割された際には遷移が増える場合も生じる．この点では抽象化を行う条件が緩く，状態の削減効果は少ない．しかしながら，提案手法の抽象化に置いて到達不可能性が削減されることはない．提案手法では，抽象化の方向性を決定する際に状態数を削減することとし，遷移内容と状態の動作の抽象化については状態数を削減する事は目的としていない．

一方従来のやり方では，抽象化の際に不具合として検出されるべき状態とその状態への遷移も削減してしまうことがある．これは抽象化後のモデルにおいて経路が縮退してしまい，抽象化前のモデルに包含されてしまうことが原因である．

今後の課題として状態数を減らすなどの抽象化の手法が必要ではあるが，提案手法を用いることで従来のやり方で生じている問題を解決する事ができる．

5.3 提案手法全体を通して

提案手法は，抽象化を対象システムについての知識を基に始めることが出来る．また，現段階では到達不可能性の検証についてであるが，モデルの持つ性質を維持したままの抽象化が可能である．データマッピングや述語抽象などの抽象化手法を適用する事ができず，どの性質・状態に着目して良いか分からない場合に置いて，抽象化のきっかけとして対象システムの動作について考えるという着目点を与え，続いて検証に必要な性質を維持しながら抽象化を行うことが出来る手法である．

6. ま と め

本論文では，形式手法の一つであるモデル検査において適用されている抽象化の手法を紹介した．そしてそれらが適用出来ない場合でも用いることのできる抽象化手法を提案した．提案手法では抽象化を 2 つの段階に分けている．最初の段階では検証対象のシステムに関する知識があれば抽象化の方向性を定めることができる．続いて遷移内容と状態の持つ動作に対して状態が抽象化後に分割されたのか，統合されたのかなどの分岐条件から抽象化を手続き的に行うことができる．

この提案手法を用いた事例として自動車のトランクなどに付いている電動クローザシステムを扱い，そこから得られた知見を考察した．これにより，課題として挙げた抽象化を行う状態・遷移に対する着目点を得ることと，検証に必要な性質を維持しながら抽象化を行うことのできる手法を提案する事ができた．

7. 今後の課題

組み込みソフトウェアの分野では，時間を考慮した状態遷移や，状態遷移に優先度付けを行っている状態遷移図も存在している．これらの概念について今後対応していくことで，今回提案した手法が扱うことのできる条件の範囲が広がるものだと考える．また，提案した手法の中で用いている遷移の抽象化方法を深めていく必要もある．抽象化する際には全ての情報を状態の抽象化に応じて用いている．そのため到達不可能性を検証する事はできるものの抽象度が高く，抽象化前のモデルで到達できない場合でも抽象化後のモデルが到達してしまう場合が考えられる．より適切な抽象化の方法を考えるべきである．

また，提案手法を用いる際には，検証対象のシステムに関する知識があることが前提となっている．この知識の活用方法次第では，様々な抽象度，性質に合わせた抽象化を行うことができる．一方で，検証対象となるシステムの知識が不足している場合は提案した手法を適用する事は難しい．そのため，知識が不足している場合でも適用でき，抽象度を選択できる仕組みを状態遷移図を記述する時点で組み込む記述方法を考案する必要がある．そうすることで，検証対象のシステムに関する知識をより有効に利用する事ができると考えている．

謝辞 この論文を執筆するのに際し，本論文で提案した手法の適用事例として扱う題材を提供して頂いた，アイシン精機株式会社 鈴村延保 様に感謝致します．

参 考 文 献

1) 経済産業省：組込みソフトウェア産業実態調査報告書 (2009).
2) 佐原　伸：〜ソフトウェアトラブルを予防する〜形式手法の技術講座，ソフト・リサーチ・センター (2008).
3) 栗田太郎：仕様書の記述力を鍛えるモバイル FeliCa 開発における形式仕様記述手法の導入事例，日経 ELECTRONICS，Vol.945, pp.133–152 (2007).
4) 中島　震：SPIN モデル検査，近代科学社 (2008).
5) 田辺良則，高井利憲，高崎孝一：抽象化を用いた検証ツール，コンピュータソフトウェア，Vol.22, No.1, pp.2–44 (2005).
6) 中島　震：Event-B デザインのモデル検査における抽象化，情報処理学会研究報告，Vol.55, pp.81–88 (2008).
7) 管籐　真，瀬口俊和，山根　智：プログラムスライシングによるプログラムのモデル検査手法，電子情報通信学会技術研究報告．SS，Vol.101, pp.17–24 (2001).