モデル検査における健全性を保持する変数抽象化方法の提案

M2010MM020 加藤 友章

指導教員 青山 幹雄

1. はじめに

近年，組み込みソフトウェアは大規模化，複雑化する傾向があり，従来のテストのみでは高信頼性の確保が困難になっている．その解決手法としてモデル検査が必要とされている．しかし，モデル検査は状態爆発問題によって，実用時間内で検証できない可能性がある[2].

本研究ではモデル検査の状態爆発問題を軽減するため，モデルの変数の抽象化する方法を提案する．

2. 問題の背景

モデル検査には対象システムのオートマトンが大規模化，複雑化することで状態爆発問題が起こり，実用的な時間で検証できない可能性がある．

状態爆発問題の軽減のため，モデルの抽象化が必要とされている．しかし，抽象化はモデルの検証したい性質に影響を与え，本来の検証結果が得られない可能性がある．

3. 抽象化

モデル検査の状態爆発問題を解決する手法として，よく用いられるものに抽象化がある[1][3]. 抽象化とはシステムから検証したい性質と無関係な情報を省略したシステムを構築することである．省略された情報は検証できないが，検証時間は抽象化前のシステムより短くなる．ここで，抽象化前のシステムを具体システム，抽象化後のシステムを抽象システムと定義する．モデル検査の代表的な抽象化に，下記に示す状態の抽象化と変数集合の値域の数を減らす変数の抽象化がある．

(1) 状態の抽象化
　不要な状態の除去や複数の状態の統合を用いることで状態数を減少させる抽象化

(2) 変数の抽象化
　大きい変数集合を小さい変数集合に写像することで，変数集合の値域の数を減少させる抽象化

本稿ではこれらの抽象化から，組込みソフトウェアの特性に対して必要な抽象化を選択し，その抽象化の方法を提案する．

4. 関連研究

4.1. 抽象化と精密化による実時間モデル検査の改善

抽象化と精密化を用いることによってモデル検査を改善する方法が提案されている[5]．ここで，変数を除去するなど，高い抽象度の抽象化を高レベルの抽象化，より具体システムに近い抽象度を低レベルの抽象化として定義する．

提案方法は，次の手順でおこなわれる(図 1).

(1) 高レベルで抽象化した抽象システムを構築する．
(2) 抽象システムでモデル検査をおこない，性質を満たせば，具体システムでも保証される．性質が満たされない場合，反例を検査し，具体システムで対応する反例が見つかれば具体システムは性質を満たさない．
(3) 具体システムで対応する反例が存在しない場合，抽象システムにおいて，その反例を引き起こす遷移を除去して，上記の(2)に戻る．
(4) 遷移の除去が出来なかった場合，精密化をおこない抽象化のレベルを低くした，新しい抽象システムを構築し上記の(2)に戻る．

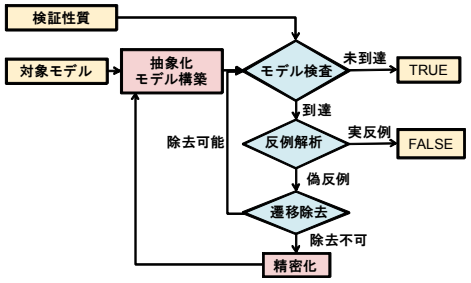

図 1 抽象化と精密化による方法

この関連研究では最初に，高レベルで抽象化を行なっているが，高レベルの抽象化は本来起こり得ない遷移によって偽反例が出力される可能性がある．また，反例が出力された場合の処理が煩雑になる．これらより，反例解析及び修正に時間を要する．

この問題に対し，抽象モデルの構築において，偽反例が出力を抑えるように適切な抽象化をすることで，偽反例の出力による煩雑な処理を回避することができる．

4.2. SPIN

モデル検査をするツールに SPIN がある[2][7]. これは，オートマトン理論と時相論理に基づいた実用的なモデル検査ツールである．

5. アプローチ

本章では抽象化方法の選択と抽象化の健全性を保持する条件を記述する.

5.1. 抽象化方法の選択

本稿では組込みソフトウェアへのモデル検査の適用を目的としている. 組込みソフトウェアはリアルタイム性を検証するためにクロック変数を, またセンサを通して速度や温度等の多様な変数を取り扱う必要がある. SPIN 等のモデル検査器を用いてそれらをモデル化した場合, 構築したモデルの検証時間は非常に大きくになってしまう可能性がある. それにより, 変数の抽象化を行う必要がある.

5.2. 抽象化の健全性

健全性とは, 抽象化の前後においてモデルが検証したい性質を保持しているという性質であり, これを満たさない場合, 検証したい性質が保証できない可能性がある. モデルの抽象化の前後で健全性を保持するために必要な条件は下記の三つである.

(1) 具体システムのモデルにおける初期状態を抽象化後のモデルに写像した状態は、抽象システムのモデルの初期状態集合に含まれる.

(2) 具体システムのモデルにおいて遷移関係が存在する二つの状態は抽象システムのモデルにおいても遷移関係が存在する.

(3) 抽象化前の任意の状態で成り立つ命題は, 抽象化後のモデルに写像した状態で成り立つ命題と一致する. ここで, 命題とはモデルの状態遷移の条件である.

6. 健全性を保持した変数抽象化方法

6.1. 健全性を保持した変数抽象化方法概要

健全性を保持する抽象化プロセスを提案する. 提案方法は図 2 に示すように, モデルの抽象化を変数集合の分割, 抽象化, 再統合の三つのプロセスで行う. 各プロセスはそれぞれ 6.2 節, 6.3 節, 6.4 節で解説する. また, 定義 1 において, モデル検査に用いるモデルの変数及び変数集合を定義する.

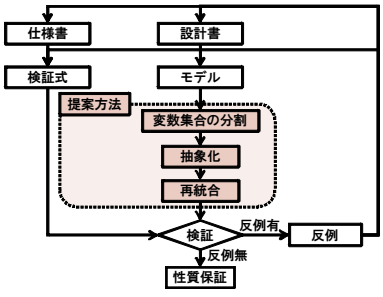

図 2　検証のプロセス

定義 1(モデルの変数及び変数集合の定義)

V :変数集合

v_i :V に含まれる変数

V' :抽象化後の変数集合

v_j' :V'に含まれる変数

AP:遷移条件を構成する命題の数

6.2. 変数集合の分割

本プロセスでは, 変数集合を分割する. 分割は遷移条件を構成する命題の真偽に基づいて以下の手順に基づいて行う. それにより, 変数は図 3 に示すように分割される.

(1) V を命題の真偽に基づいて分割を行い, 2^{AP} 個の変数集合 V_i に分割する. 健全性を保持する条件を満たすために, 命題真偽が異なる変数同士は異なる変数へと写像しなければならない. そのため, 分割前後の変数集合が $V=\Sigma V_i$ 及び $V_i \wedge V_j = \varnothing$ を満たしていることを確認する必要がある. また, 変数集合に対して二つ以上の命題が存在し, 一つ以上の命題からそれ以外の命題の真偽が決定するとき, ∅となる変数集合が存在する.

(2) 分割後の各変数集合の値域を用いることで, その変数集合の地域内における検証ができる. 上記の(1)において∅となった変数集合は, 具体モデルにおいて起こらないため, 検証する必要はない.

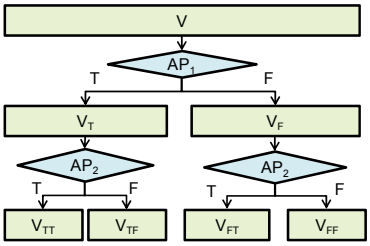

図 3 変数集合の分割

6.3. 抽象化

本プロセスでは, 6.2 節で分割した変数の抽象化を以下の手順に基づいて行う. それによって抽象化の前後の変数は図 4 に示すように抽象化される.

(1) 分割した変数集合が∅以外の場合, それぞれ変数集合を一つの変数に写像し, 抽象化する. 抽象化後の変数は v_j'であり, 抽象化前の変数集合に含まれる全ての変数と同一の命題真偽を持つように定義する. ここで, 具体システムの初期状態となるいずれかの変数が V_i に含まれる場合, V_i から写像した変数 v_i'は抽象システムにおける初期状態とする.

(2) 分割した変数集合が∅の場合, その変数集合を削除する. これによって, 分割した変数集合のうち∅となった変数集合の数を x 個とすると, 変数集合を 2^{AP}-x 個の変数に写像できる.

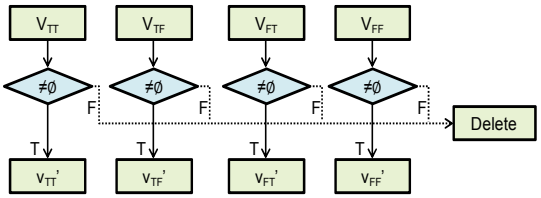

図 4 変数の抽象化

6.4. 再統合

本プロセスでは以下の手順に基づいて 6.3.で抽象化した変数を統合し，一つの変数集合とする．その変数集合を具体システムの変数集合を置き換えることによって抽象システムを構築できる．変数は図 5 に示すように統合される．

(1) 抽象化した変数を一つの変数集合として統合する．統合された変数集合を V'として，$V'=\Sigma v_j'$を満たすように統合する．また，V_iの値域に含まれるいずれかの変数が，V_jの値域のいずれかの変数へ状態遷移が可能な場合，それらの変数集合の写像した変数 v_i'，v_j'において，v_i'からv_j'への状態遷移が可能とする．

(2) 統合した変数集合 V'を用いた抽象システムを構築し，モデル検査をする．

抽象システムの変数集合は具体システムの変数集合と同等の命題真偽を持つ．また，各状態遷移を引き起こす条件は命題によって与えられる．そのため，抽象化前に可能だった状態遷移は，抽象化後も遷移可能となる．そして 6.3 節より，具体システムの初期状態は抽象システムの初期状態に含まれる．これらより，構築した抽象システムは健全性を保持する．抽象システムを用いたモデル検査によって抽象化前の全ての変数を含むモデルの性質を保証できる．

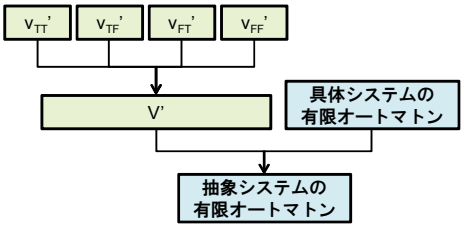

図 5 変数の再統合

7. 例題への適用

提案方法の評価するため，自動車の CCS(Cruise Control System)及び ABS(Antilock Brake System)の二つのシステムへ適用した．検証にはモデル検査ツール SPIN 及びその UI である XSPIN を用いた．モデルは Promela で記述し，検証式は LTL 式で定義した[7].

7.1. CCS への適用

CCS はアクセル操作なしに自動車を定速走行させるシステムである．CCS の解説書を用いてモデルと検証式を構築した[4]．モデルの状態遷移図を図 6 に示す．

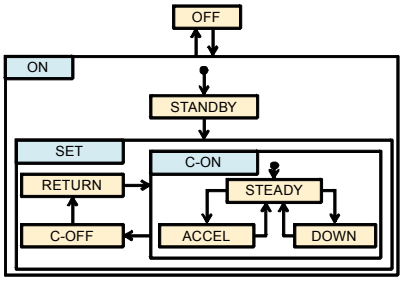

図 6 CCS の状態遷移図

検証式は解説書より CCS 動作上において必ず保証されなければならない四つの検証性質を用い，それぞれ CCS-PROPERTY1,CCS-PROPERTY2,CCS-PROPERTY3, CCS-PROPERTY4 として定義する．

7.2. ABS への適用

ABS は自動車が急ブレーキを踏んだ場合などに車輪がロックすることを防ぐポンピングブレーキを自動的に行うシステムである[6]．ABS の構造書を元にモデルを構築した[3]．ABS は車速が 0 になるまで四つのモジュールが並列して動作する．各モジュールの状態遷移図を図 7 に示す．

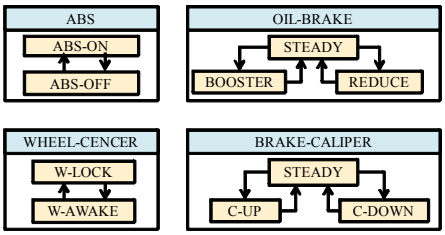

図 7 ABS の状態遷移図

検証式は構造書より ABS の動作においてユーザへの危険を避けるために重要な三つの性質を形式化し，ABS-PROPERTY1,ABS-PROPERTY2,ABS-PROPERTY3, として定義する．

7.3. 評価基準

評価基準は「健全性を保持した抽象化」ができているかと「モデル検査のリードタイム短縮」によって評価する．

リードタイムとは着手から全ての工程が終了するまでの所要時間として定義する．モデル検査のリードタイムには「モデルと検証式を構築する時間」，「検証時間」，「反例解析をする時間」の三つが含まれる．モデルの大規模化，複雑化に伴い「検証時間」は著しく増加する．また，「反例解析をする時間」は反例に至るまでの状態遷移の数によって著しく増加する．よって，提案方法による抽象化前後でこれらを比較する．

7.4. 提案方法による抽象化前後の検証時間の比較

構築したモデルに対し，提案方法による抽象化を行い，

同一の検証式と抽象化前後のモデルを用いてそれぞれ100回ずつ検証時間(ミリ秒)を測定した. この結果を変数の値域の数と状態数, 出力反例までの遷移数(個)と共に比較する. そして, CCS 及び ABS に対して CCS-PROPERTY1, ABS-PROPERTY1 を検証式として用いた場合の出力結果を表1及び表2で示す.

表 1 CCS の抽象化前後の比較(個, ミリ秒)

	変数	状態数	遷移数	Ave	Dev
抽象化前(A)	121	148,054	2,375	137.1	2.5
抽象化後(B)	3	1,738	87	21.1	1.2
比率(B/A)	1/40	1/85	1/27	1/6.4	1/2.1

表 2 ABS の抽象化前後の比較(個, ミリ秒)

	変数	状態数	遷移数	Ave	Dev
抽象化前(A)	2,121	274,825	1,717	352.5	5.2
抽象化後(B)	9	1,026	190	23.9	1.2
比率(B/A)	1/236	1/270	1/9	1/15	1/4.3

8. 考察

8.1. 健全性の保持

抽象化前後のモデルの比較より, 抽象化前の初期状態は抽象化後も初期状態に含まれること, 抽象化前後で変数の命題真偽が同一であることを確認した. また, 状態遷移の条件は命題となっており, 命題が同一の場合, 同一の状態遷移が可能となる. これらより, 健全性を保持する三つの条件を満たしていることを確認した.

8.2. 抽象化前後の検証時間削減

抽象化前と比較して CCS, ABS のモデルに含まれる変数集合は 1/40, 1/236, 検証時間は 1/6.4, 1/15 となった. CCS, ABS のいずれの場合においても, 検証時間は変数集合の値域の数ほど減少していない. これは, 検証時間には反例の出力, 書き込み時間等, 抽象化によらず必要とする時間が含まれることが要因と考えられる. しかし, 提案方法を用いた抽象化によって検証時間が大幅に短くなっており, モデル検査のリードタイム短縮に効果があると言える.

8.3. 抽象化前後の出力反例の遷移数

出力反例は初期状態から検証性質に反するまでの状態を示すため, 状態数の減少に関係して短くなると考えられる. 適用結果では状態数が 1/85, 1/270 に対し, 出力反例は 1/27, 1/9 となった. CCS 及び ABS のいずれの場合においても, 状態数, 出力反例は減少しているが, 出力反例は状態数ほど減少していない. これは, 車速の加減速や減速値という複雑さの少ない変数集合のみの抽象化が要因と考えられる. しかし, 出力反例までの遷移数は抽象化前後で大幅に減少しており, 容易となると言える.

8.4. CCS, ABS 抽象化前後の減少割合の差異

CCS, ABS の抽象化前後の比較において, 出力反例までの遷移数の減少割合は大きく異なった. これは, CCS の車速は設定車速との差異にあわせて加減速し, 特定の車速を条件とした CCS の状態遷移が多い. ABS の車速は減速値に基づいた減速の一方向変化であり, 車速を遷移条件とするABSの状態遷移はCCSと比較して少ないことが要因と考えられる. よって, 本提案方法では, 変数の複雑度が反例までの遷移に影響を与えやすいと考えられる.

9. 今後の課題

提案方法では, モデルに含まれる命題真偽に基づいた抽象化という明確な基準が存在する. 今回の適用において, 抽象化は手作業で行なっているが, 具体システムのモデルを抽象システムのモデルへと自動変換するツールを構築することで, より正確で効率的な抽象化が可能と考えられる. また, ツール作成においては変換に用いる時間を考慮したアルゴリズムを構築する必要がある.

10. まとめ

本稿では, モデル検査の組込みソフトウェアの特徴からモデル抽象化の方法として変数の抽象化に着目し, モデルの健全性の保持条件を導入することによって, 命題に基づく変数抽象化方法を提案した. また, CCS, ABS の例題を用いた適用により, 健全性の保持条件を満たしていること, リードタイムを短縮できていることを確認した.

参考文献

[1] 青木 翼, ほか, UPPAAL によるモデル検査適用ガイドラインの作成(解析・検証), 情報処理学会, ソフトウェア工学会研究報告, Vol. 2008, No. 29, pp. 203-210.

[2] M. Ben-Ari, Principles of the Spin Model Checker, 2008.

[3] 石黒 正揮, ディペンダブルシステムのための形式手法実践ポータル, 2011, http://formal.mri.co.jp/.

[4] MathWorks, クルーズコントロール制御編-サンプルモデル解説書-, 2011, http://www.mathworks.co.jp/products/stateflow/technicalliterature.html.

[5] 中島 一, 亀山 幸義, 抽象化と精密化による実時間モデル検査の改善, 情報処理学会論文誌, Vol. 45, No. SIG12 (PRO 23), 2004, pp. 11-24.

[6] 日本自動車整備振興会連合会, アンチロック・ブレーキ・システム, 2011, http://www4.jaspa.or.jp/jaspahp/user/publication/pdf/teisei_chassis1abs_kodoseibi.pdf.

[7] Spin – Formal Verification, 2011, http://spinroot.com/spin/whatispin.html.