モデル検査によるリアルタイムオペレーティングシステムの検証実験

青木利晃†

北陸先端科学技術大学院大学 安心電子社会研究センター†

1. はじめに

組込みソフトウェアで用いられるリアルタイムオペレーティングシステム(RTOS)には，ソフトウェア実行の共通基盤として用いるため，高い信頼性が求められる．一方で，パフォーマンス向上の目的から，内部操作をインタリーブさせるなどして，複雑な実装がされている場合が多い．このような実装の正しさの確認は非常に困難であり，実際，それが原因で誤りが発生している．そこで，我々は，モデル検査手法を用いて RTOS の振る舞いを検証する手法について研究を行っている．対象はμITRON 仕様の RTOS で，モデル検査ツール Spin を用いている．

2. 検証対象と検証モデル

我々は，これまでに，μITRON 仕様の RTOS を Spin 上で模倣する RTOS ライブラリを Promela により実装した[1]．このライブラリは，RTOS 上で動作するマルチタスクを検証するためのものであるが，一方で，RTOS 自体を Promela によりモデル化したものとも見なせる．そこで，このライブラリを検証対象の RTOS のモデルと見なして検証実験を行っている．以降，この検証対象のことを RTOS ライブラリではなく，**RTOS モデル**と呼ぶことにする．

RTOS モデルは，RTOS と同様，それ自体では動作しない．タスクやその他のプログラムによるサービスコールの呼び出しや割り込みが発生して動作する．よって，RTOS モデルを検証するためには，まず，RTOS を動作させる部分を作成する必要がある．また，検証する性質についても厳密に定義しなければならない．本研究では，RTOS モデルがμITRON 仕様に準拠していることを検証することにしたが，μITRON 仕様は自然言語で記述されているため，仕様にもとづいて Spin で検証可能なくらい形式的に性質を定義する必要がある．さらに，このような RTOS を動作させる部分と検証する性質は一緒に考えなければならない．検証したい性質によって，RTOS の動作のさせ方が異なるからである．

これらの動作や性質は，検証目的に基づいて適切に作成しなければならない．すなわち，モデル化が必要である．

本論文では，このモデルのことを**検証モデル**と呼ぶことにする．

3. 検証モデルの構築

μITRON 仕様は機能毎に分けられて記述されているが，そのような機能に横断的な動作で誤りが発生する場合が多い．例えば，タスクの状態変化と割り込み発生と共有資源操作が連続で発生する場合などである．このような横断的な動作を，直接，人間がモデリングすることは避けるべきである．考え漏れが生じやすいからである．横断的な動作の組み合わせを作成するのはモデル検査ツールに任せるべきである．よって，人間が機能毎の動作をモデル化し，その機能横断的な動作をモデル検査ツールで網羅的に探索できるような検証モデルを構築する必要がある．

そこで，まず，タスクのライフサイクルを操作する機能に注目して，検証モデルを作成した．この検証モデルでは，正常なサービスコールの呼び出し，すなわちエラーを返さない任意の呼び出し列のモデル化を行った．このような呼び出し列を実現する状態モデルを，仕様から直接作成するのは困難である．そこで，まず，サービスコール毎に，呼び出すことのできる事前条件，事後条件を形式的仕様記述言語Ｚで定義した．図１はその記述の一部である．変数 state は，サービスコールが呼び出される前後で，対象となるタスクがあるべき状態を表現している．μITRON では，slp_tsk の呼び出しはキューイングされる場合があるので，それを数えるカウンタ wupnum を導入して，状態の変化をモデル化した．このＺ記述から state 変数に基づいて状態遷移モデルを作成することができる．そして，それぞれの状態において，RTOS 上の対象タスクが期待する対応する状態になっているかどうか検証する．このようにして，タスクのライフサイクルに関しては，検証モデルを作成できた．

次に，セマフォについても同様に検証モデルを作成した．検証モデルは，RTOS の内部状態を詳細に反映する必要はない．詳細にモデル化しすぎると，最終的には，RTOS のモデルと同じものになってしまう．検証目的に応じて，適切な抽象化を行うべきである．RTOS モデルではセマフォによりブロックするタスクを格納するブロックキューを

A Verification Experiment of Real-Time Operating Systems with Model Checking

†Research Center for Trustworthy e-Society, Japan Advanced Institute of Science and Technology

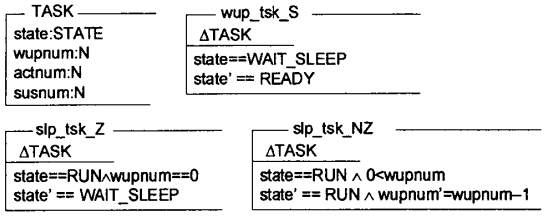

図１：Ｚによる記述

用いているが、セマフォの検証モデルではそれを用いずに，sig_sem や wai_sem を呼び出した回数を数えるカウンタを用いてモデル化を行った．

4. 単体検証と複合検証

構築した検証モデルは，それらの各々について検証を行う場合と，組み合わせて検証する場合が考えられる．前者の場合を**単体検証**，後者の場合を**統合検証**と呼ぶことにする．単体検証では，それぞれの機能の観点から，複合検証では，複数の機能に横断的な観点から正しさを保証することになる．単体検証を実行したところ，いくらかの誤りを発見できた．例えば，特定
の検証モデルの状態遷移のパスにおいて優先度キューの操作を誤りや，条件の考え洩れなどである．

複合検証では，機能毎の検証モデル間の以下のような関連が問題となる．

- 検証モデルの状態同士の関連．タスクが RUN 状態の時しかセマフォの操作ができない，など．
- 検証モデルの構成に関する関連．タスクが持っているセマフォの数，セマフォを利用しているタスクの数，など．

そこで，図２のようなクラス図により，検証モデルと検証対象の関係をモデル化した．まずは，検証モデルの N 対 M 関連の取り扱いを考える必要がある．モデル検査では，無限の数は取り扱え無いため，検証モデルの組み合わせは有限にする必要がある．よって，N,M の値を有限の範囲で組み合わせて検証することが考えられる．一種の有界モデル検査である．

状態同士の関連については，それぞれの状態を参照して，厳密に実現する方法や，関連は無視して検証する方法が考えられる．後者の場合は，より広い振る舞いに対して検査することになるので，無意味なエラーが検出されるかもしれない．前者については，状態の参照関係を厳密に記述する手法，および，それに基づいて Promela のモデルを作成する手法が必要である．

現在までに，複合検証に関しては，タスクとセマフォを１対１，１対 2,2 対１の関係で，互いの状態を参照して厳密に実行列を実現する手法で検証を行ったが，誤りは発見されなかった．

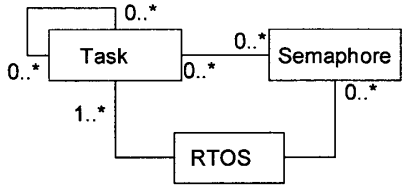

図２：検証モデルの関連

5. 考察とまとめ

モデル検査により検証を行うには，検証対象の振る舞いだけでなく，検証する動作列や性質についてもモデル化する必要がある（検証モデルの作成）．これはテスト手法におけるテスト仕様やテストケースの作成と同様のものであるが，モデル検査を適用することを目的としている点で異なる．この検証モデルは，モデル検査の特性を考慮して作成すべきである．１つの有効な適用法は，人間が苦手とする仕様横断的な性質を検証することである．そのためには，仕様と直接対応可能な検証モデルの作成法を提案する必要がある．さらに，検証モデルを組み合わせることにより，仕様横断的な性質を検証可能である必要もある．このような検証モデルの分離・合成を実現するために，２つの仕組みを提案すべきである．１つは，抽象化法である．検証モデルで必要な詳細は，目的となる性質を検証できる程度で良い．よって，適切に抽象化を行うことにより，検証モデル間の依存関係を断ち切ることができる場合がある．例えば，タスクとセマフォの検証モデルのようにカウンタを用いて単純なモデル化を行えば、それらの間の依存関係を小さくすることができる．２つ目は，合成可能な範囲で明示的に依存関係を記述することである．依存関係の記述は，ある程度，独立して考えられなければならない．複数の検証モデルの中身を詳細に見なければならないのであれば，仕様横断的な性質を人間が考えているのと同じだからである．このような依存関係の記述は，アスペクト指向などの考え方が応用できるのではないかと考えている．

現在は，検証モデルに対応する Promela 記述をアドホックに作成している．系統的な検証を実現するためには，以上のような検証モデルの作成法，および，それらに基づいた単体検証と複合検証手法を提案する必要があり，現在，研究中である．

参考文献

[1] 青木利晃，片山卓也：RTOS に基づいたソフトウェアのためのモデル検査ライブラリ，組込みソフトウェアシンポジウム 2005，pp. 56-63，2005.