# 並行ソフトウェアの実行前検査に関する研究

2004MT073 小笠原 末季
指導教員　野呂 昌満

## 1　はじめに

並行ソフトウェアでは，実行時の挙動を把握するのが難しく，システムの欠陥 (fault) が発生しやすい．これは，設計段階でソフトウェアの設計に誤りがあることが一つの原因である．設計の誤りは実装後実行されるまで発見されにくく，発見されると大幅な手戻り作業が発生しやすい．それゆえ設計段階に実行前検査をおこなうことが必要とされている．実行前検査をおこなう方法の一つとして，モデル検査ツールの利用がある．モデル検査ツールを用いるためには，ソフトウェア設計を形式記述やモデル検査言語に書き換える必要がある．しかし，広く用いられている記法にしたがって記述された設計からモデル検査言語への作成に変換するのに多大な手間がかかるので，実行前検査を実用化するのが困難となっている．本研究の目的は，オブジェクト指向ソフトウェア開発を前提として，実行前検査を実用化することである．モデル検査言語の記述の手間を省くことで開発者の支援をおこなう実行前検査支援ツールを提案する．支援ツールは，設計段階の UML 図をモデル検査言語へ対応づけ，モデル検査ツールの入力コードを自動生成する．

本研究は次のように進めた．

- 検査項目，入力対象，使用するモデル検査ツールの決定
- 実行前検査支援ツールの設計と実現
- 実行前検査支援ツールの実用化についての考察

## 2　並行ソフトウェアの実行前検査方法

本研究では実行前検査を，設計仕様書に対してモデル検査をおこなうことと定義する．本節では検査に用いるツールの決定や，検査すべき項目を決め，入力となる設計仕様書から検査コードの生成方法について説明する．

### 2.1　検査項目

検査項目にはデッドロックおよびスタベーションなどの並行プログラム特有の検査と，システムが仕様を満たしているかの検査が考えられる．デッドロックおよびスタベーションは，並行プログラムにおいて発生するか否かが分かりにくく，大きな問題となっている．本研究では並行プログラムの主要な問題であるデッドロックおよびスタベーションの解決を目的とする．よって検査項目はデッドロックおよびスタベーションとする．また，使用する図のセマンティックスは本研究ではチェックしない．

### 2.2　入力対象

本研究では，並行プログラムを並行して動く状態遷移機械の集合とみなし，それぞれを UML のステートマシン図で表現することとした．ただし，ステートマシン図には，そのアクション記述が一般に自然言語で記述されて，複数の状態遷移機械間におけるイベントのやりとりがあいまいであるという問題がある．モデル検査をおこなうためには，アクションで発生するイベントやイベントの送り先を明示しなければならず，ステートマシン図のみではアクションについての詳細なこれらの情報が不足している．アクション部分の不足する情報を補う手段として，形式記述による方法と，他の UML 図 (シーケンス図) による方法がある．形式記述による方法では，ユーザが UML 図に則らない記述を要求され，また形式記述に習熟するコストも必要となる．このため本研究では他の UML 図 (シーケンス図) による方法を採用した．

### 2.3　モデル検査ツールの比較検討

並行プログラムのモデル検査ツールとして代表的なものに FDR[3] と SPIN[2] がある．FDR は CSP 理論に基づいたモデル検査ツールで，検査言語は CSP[1] である．SPIN はチャネル通信オートマトンのためのモデル検査ツールで，検査言語は Promela である．

**モデル検査ツールの比較検討**

本研究で使用すべきモデル検査ツールを二つの観点から比較した．第一の観点は，ユーザにとってエラーメッセージが分かりやすいか否か，第二の観点は，自動生成ツールを作成するさいに，UML 図とモデル検査言語との対応づけがしやすいか否かである．

- エラーメッセージの表示方法
  - FDR : システムが各プロセスを木構造で表示
  - SPIN: シーケンス図に類似した図で表示
- UML 図とモデル検査言語の対応づけ
  - FDR : シーケンス図との対応 ◎
    ステートマシン図との対応 △
  - SPIN: シーケンス図との対応 ○
    ステートマシン図との対応 ◎

エラーメッセージの分かりやすさではシーケンス図に類似した図でシミュレーション結果を表示する SPIN の方が分かりやすいと判断した．また，モデル検査言語の対応づけでは，SPIN ではステートマシン図と一対一対応になっており，シーケンス図との対応も難しくない．二つの観点からともに SPIN の方が優れていると判断したので，使用するツールは SPIN とする．

## 3　支援ツールの設計と実現

ステートマシン図，シーケンス図は UML ツールを用いて記述する．UML ツールから出力された情報は，われわれの研究室で開発されたツールにより中間表現に変換される．自動生成ツールは中間形からからモデル検査コードへ対応づける．ツールの全体像を**図 1** に示す．

**UML 図と Promela の対応**

UML 図をもとに，検査するために生成される Promela コードは大きく 4 つの部分から構成される．イベント

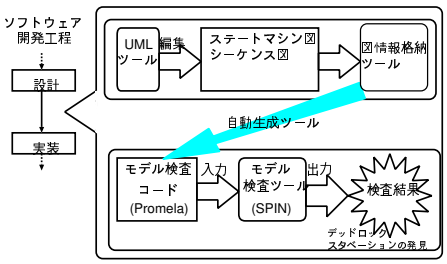


図 1　本研究による支援の全体像

宣言部分，状態遷移機械の振る舞いを記述する部分，状態遷移機械にランダムに初期イベントを送信し動かす部分，状態遷移機械のインスタンスを生成する部分である．これを Promela 定型コードと呼び，本研究の自動生成対象とする．ここで UML 図の情報と対応しているのは，イベント宣言部分と状態遷移機械の振る舞い部分である．状態遷移機械の振る舞いについて UML 図の情報と Promela の対応を**図 2** に示す．

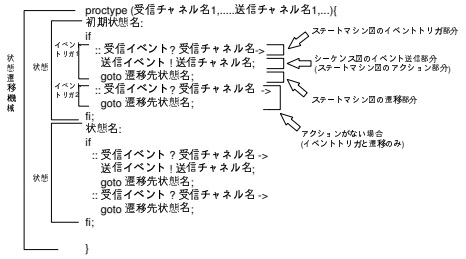


図 2　状態遷移機械の振る舞い部分

## 4　考察

本研究で提案した支援方法の実用性について考察する．本研究で提案した Promela 定型コードを用いて，UML 図から手動で対応づけした Promela コードで検査をおこなった．例として，哲学者の食事問題とプリンタスキャナ問題をもちいた．

### 4.1　提案した支援ツールの成果

本研究で扱った例の結果を述べる．哲学者の食事問題では，デッドロックが発生し，正しい結果が得られた．プリンタスキャナ問題でも，同様の結果が得られた．さらに，二つのプリンタスキャナ問題を一つのシステムとして扱い，一方のシステムがデッドロックし，もう一方は正常に動くシステムについても検査をおこなった．その結果，システム全体としては正常だと判断された．Progress ラベルを適切な箇所に付与したところ，異常 (スタベーション) と正しく判断された．以上のことから，本研究で提案した支援方法の成果と，問題点をあげる．成果として，本研究で提案した変換規則にしたがって，UML 図から Promela コードへ機械的に変換することができた．支援ツールを作成することで，実行前検査の問題であった手間を省くことができる．また，機械的に変換したコードはデッドロックの有無を検査できる．以下に問題点を述べる．

- スタベーションの検査ができない
- 正常終了しているシステムもデッドロックと判断する場合がある
- SPIN の検査結果を基にユーザがエラー原因を特定するのが困難である

### 4.2　問題点の解決策

**ラベルを取り入れたツールの作成について**

スタベーション検査をするためには，Promela のラベルを用いることが必要である．ラベルには Progress ラベルと End ラベルがある．Progress ラベルは，必ず実行して欲しい場所に “ Progress: ” というキーワードを記入することで，必ず一度は実行しているかというスタベーションの検査に用いることができる．また，End ラベルでは，正常終了してよい箇所に “ End: ” というキーワードを記入することで，その場所で停止しても正常と見なされる．そのために，正常終了して欲しい箇所でシステムが停止した場合，ラベルを付与する場合には，デッドロックの誤検出を回避することができる．本研究で提案したツールにラベルを取り入れる方法を考察する．まず，ユーザが UML 図にラベルを記入する箇所を記述する方法が必要である．ラベルは細かな振る舞いについて付与するのが適当なので，シーケンス図に記述するのがよい．シーケンス図では，イベントを受信前，送信前，送信後とラベルをつけるタイミングを細かく分けることが可能である．

**エラーフィードバックツールの支援**

本研究の提案するツールでは，モデル検査をおこなった後，ユーザはモデル検査ツールのエラーメッセージを読み取るために，モデル検査の出力を知らなければならない．また，モデル検査の出力を確認し，それらと UML 図との対応を理解する必要がある．よって，このような手間を省くためには，モデル検査の出力を理解せずに設計図の改良を図るためにエラー原因を UML 図にフィードバックするツールの作成が有効であろう．

## 5　おわりに

本研究では，実行前検査の実用化支援のために，UML 図からモデル検査コードを自動生成する方法の提案をした．また，支援ツールの設計をおこない，実現に向けた課題の整理をした．今後の課題として，提案した支援ツールの実現があげられる．また，現在提案している支援ツールをふまえて，デッドロックとスタベーションのより正確な検査をしたい．ラベルを考慮した検査方法ならびにエラーメッセージと UML 図の対応関係の考察をおこなうことも今後の重要な課題である．

## 参考文献

[1] C. A. R. Hoare：Communication Sequential Process, Prentice Hall (1992).

[2] Gerard J. Holzmann：The SPIN Model Checker Primer and Reference Manual, Addison-Wesley (2004)

[3] Formal Systems (Europe) Ltd: “FormalSystems”, http://www.fsel.com, Sep. 2007.