社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－ＳＥ－156 (1)
2007－EMB－ 5 (1)
2007／5／28

モデル検査への事前条件・事後条件検証の導入

平野清美, 宇佐美雅紀, 藤倉俊幸

イーソル株式会社リサーチ＆コンサルテーションサービス部

アブストラクト:
従来の形式仕様記述のレベルはデータ制約を扱うシミュレーションまででであったが, 本研究によりモデル検査技術を利用することで検証レベルを上げることができたので報告する.

A study of an integration "design by contract" and the model checking

Kiyomi HIRANO, Masanori USAMI, Toshiyuki FUJIKURA

Research & Consultation Services Group, eSOL Co., Ltd.

Abstract:
This paper describes about the enhancement of the verification level of embedded software. Although the verification level of the conventional formal specification language is limited to a simulation, using model checking enhance the verification level.

1. はじめに

Design by Contract(以下, DBC と略記)では, 事前条件・事後条件・不変条件を静的に指定してソフトウェアの設計検証を行うことができる. 振る舞いを検証するには Z や VDM, OCL のような形式仕様記述から, Eiffel 等のシミュレーションおよびテスト環境があるが, 網羅的な検証には至っていない. 本研究によりモデル検査技術を利用することで検証レベルを上げることができたので報告する. (図 1.)

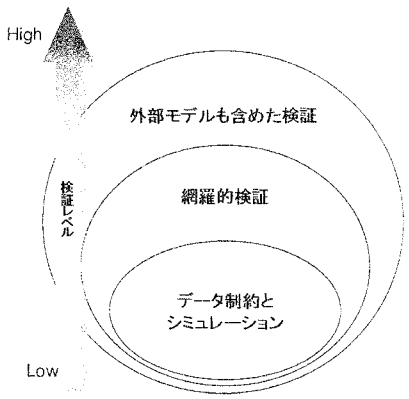

図 1. 検証レベル

2. 手法の概要

2.1. 形式仕様記述

VDM などの形式仕様記述では厳密に構文と意味が規定されており，自然言語の曖昧さを排除することができる．また，一定の条件下での入力/出力状態の関係を示す事前及び事後条件により各オペレーションを規定できるが，シミュレーションしかできない．つまり証明にはならない．従来は定理証明を用いるアプローチをとっていたが未だ成功していない．

本研究では定理証明ではなく，モデル検査ツールを用いた網羅的な検証を利用して証明するアプローチをとった．

2.2. 動作パターン生成

一般的な条件の下で DBC が成立することを証明することは困難なばかりでなく，実際には起こらない振る舞いに対する証明は不要でもある．ここでは，実際に起こりうる動作パターンをモデル検査ツールにより生成した．そして，その動作パターンに対してのみDBCが成立することを証明する．動作パターンを網羅的に生成するためにLTSAを用いた．

LTSA とはイベントやアクションの順序をいくつかの「プロセス」と呼ぶ単位で指定すると，それぞれのプロセスを任意の順番で実行した場合，全体でどのような実行のしかた（実行パス）があるかを網羅的に作り出してくれる．

2.3. DBC 検査

DBC と動作モデルを統合的に検証するために形式仕様記述と LTSA の動作モデルを NuSMV に変換した．

NuSMV で使用する SMV 言語は，LTSA が生成する有限状態遷移系を定義することができる．また，状態を変数の組で表現し，遷移関係を状態と状態の直積上の関係で与えることができる．さらに，SMV によって検証する性質を時相論理(Temporal Logic) によって与えることが出来る．この結果，単に DBC の正しさを証明するだけでなく，時相論理式による検証も可能になった．

3. サンプル

簡単なサンプルによって実際の検証手順を示す．a が 3 個と b が 2 個から合計で 3 個取り出すときの取り方は

$$\frac{3!}{1!2!}+\frac{3!}{2!1!}+\frac{3!}{3!}=7$$

で 7 通りとなる．

この取り出し方をLTSAのモデルで記述すると以下のとおりとなる．

```
A = (a->a->a->END).
B = (b->b->END).
T = ({a,b}->{a,b}->{a,b}->END).
```

このとき LTSA で生成された動作パターンは図 2. のとおりとなる．

次にこの動作パターンを変換スクリプトにより NuSMV モデルの状態マシンに変換した．その後，事前条件・事後条件を追加した．

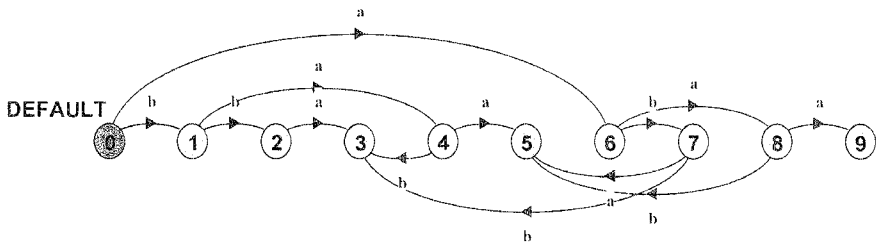

図 2. 起こりうる動作パターン

条件の記述は以下のとおり.

・変数として以下を定義する
Na : 取り出した a の数
Nb : 取り出した b の数
Ma: 残っている a の数
Mb: 残っている b の数
アクション
a : a を取り出すこと
b : b を取り出すこと

・不変条件
Na + Ma = 2
Nb + Mb = 3

・初期状態
Ma = 2
Mb = 3

・a アクション
事前条件
Ma > 0

事後条件
Na > 0, Na = Na' + 1
Ma ≧0, Ma = Ma' − 1

そして, これらの条件が図 2.の全ての動作パターンを動いた場合でも成立することを検証した.

スクリプト変換により作成した NuSMV モデルを図 3.に示す.

```
DEFINE
  pre_a := 1;
  pre_b := 1;

init(state) := 0;
next(state) := case
  state = 0 & input in { a } & pre_a : 6;
  state = 0 & input in { b } & pre_b : 1;
  state = 1 & input in { a } & pre_a : 4;
  state = 1 & input in { b } & pre_b : 2;
  state = 2 & input in { a } & pre_a : 3;
  state = 4 & input in { a } & pre_a : 5;
  state = 4 & input in { b } & pre_b : 3;
  state = 6 & input in { a } & pre_a : 8;
  state = 6 & input in { b } & pre_b : 7;
  state = 7 & input in { a } & pre_a : 5;
  state = 7 & input in { b } & pre_b : 3;
  state = 8 & input in { a } & pre_a : 9;
  state = 8 & input in { b } & pre_b : 5;
  1:state;
esac;
```

図 3. NuSMV モデル

4. まとめ

事前条件・事後条件・不変条件が成立することを一般的な定理証明ではなく, 実際に起こりうる動作パターンを生成し, その動作パターンの下で成立することを証明する手法を提案した.

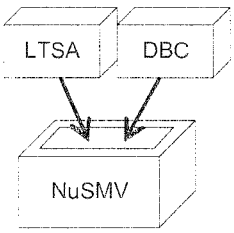

図 4. 全体の流れ

参考文献

[1] Bertrand Meyer: Eiffel: The Language, Prentice Hall, 1992.

[2] Jeff Magee,Jeff Kramer,Concurrency: State Models & Java Programs,Wiley,2006.

[3] モデル検査ツール NuSMV: http://nusmv.irst.itc.it/