B-036

拡張状態遷移技法を用いた仕様検証の実験
—組込み用 OS 試作への適用—

An experiment on the verification of specification with the extended state transition method - Application to the design of an operating system for embedded systems -

野口 健一郎† 川守田 慶†
Kenichiro Noguchi Kei Kawamorita

1. はじめに

セパレーションカーネル方式を採用した組込み用 OS を試作中である[3]。その仕様を、状態遷移技法を拡張した方式を用いて検証する実験を行った。状態遷移表に変数の記述を許した「拡張状態遷移表」による仕様記述について、その正しさを検証するシステムを開発した。それを用いて仕様検証を行った。

2. 拡張状態遷移技法について

(1) 拡張状態遷移技法とその利点

仕様記述ではセマンティクスの記述が課題である。状態遷移表（図）はそれが可能であるが、状態数の増加、無限状態への対応が不可、などの問題がある。その解決策として状態遷移表に変数の記述を許した拡張状態遷移表[2]がある。拡張状態遷移表を用いた仕様記述の利点は、

・仕様のセマンティクスを明確に記述できる。
・決定性の記述ができる。

(2) 拡張状態遷移技法の OS 仕様記述への適用例

OS の仕様は時間的順序関係が重要で、状態遷移技法はその記述に適している（プロセスの状態遷移は典型例）。拡張状態遷移表を OS 仕様記述に適用した例は[1, 2]にある。

3. 拡張状態遷移技法を用いた仕様検証システム

(1) 概要

開発した仕様検証システムの概要を図 1 に示す。

(2) XML による拡張状態遷移表の記述方式

プログラムによる処理を可能にするために、拡張状態遷移表を XML で記述する方式を確立した。

記述で用いる主な要素を表 1 に示す。

action 要素の内容として動作を記述する。これには、仕様実行で使うプログラミング言語を用いる。

(3) XML 表記拡張状態遷移表から実行可能な状態機械プログラムへの変換ツール

XML 表記から Java による状態機械プログラムへの自動変換ツール（XMLSMtoJavaSM）を作成した。XML パーサは SAX を用いた。

(4) 仕様検証の方法

仕様実行環境が、入力シーケンスを受け取り、状態機械をコールしてそれを動作させ、実行結果を出力する。実行結果には実行ログ、状態遷移履歴、変数などの事前状態と事後状態情報が含まれる。入力シーケンスと実行結果から、仕様を検証する。なお、仕様実行環境は今回の仕様検証用に作成した（汎用になっていない）。

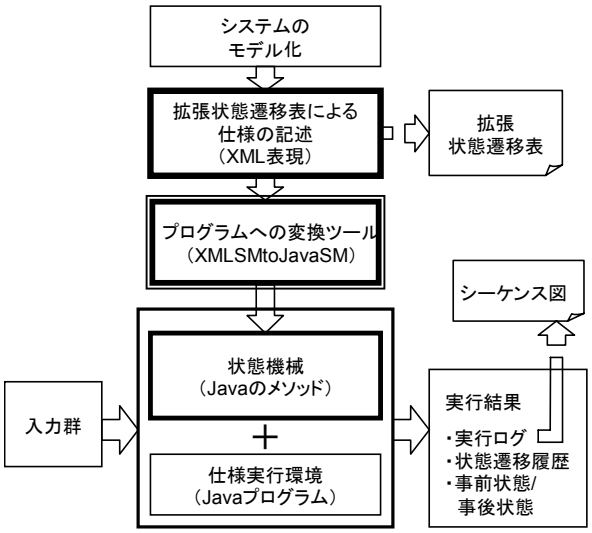

図 1 拡張状態遷移技法を用いた仕様検証システムの概要

表 1 拡張状態遷移表の XML 記述で用いる主な要素

XML 要素	説明
statemachine	ルート要素
input	入力
predicate	条件（必要な場合）
state	状態
stategroup	動作が同じ複数状態のグループ化
elementstate	stategroup 内の要素状態(空要素)
action	動作

4. 組込み OS 仕様検証への適用実験

(1) 対象とした仕様

現在セパレーションカーネル方式を採用した組込み用 OS を試作中である[3]。その主要機能としてパーティション間通信機能がある。この設計に適用した。

peer-to-peer 方式と client-server 方式の 2 種類の通信方式について実験した。後者のほうがすっきりした仕様になることが分かった。以下、後者を例として示す。

この通信方式では、次の 3 つのカーネルコール（セパレーションカーネルコール）を持つ。

send: 宛先を指定してメッセージを送り、応答を待つ。

receive: メッセージが到着していれば受け取り、そうでなければ待つ。

reply: receive したメッセージ(id で区別)への応答を返す。

なお、実行順序を変えるための yield も含めた。

(2) モデル化

仕様記述の前提として、システムのモデル化が必要であ

† 神奈川大学理学研究科情報科学専攻

る。（システムをひとつの機械として「外部的に」仕様を記述するのが理想だが、実用的なシステムではそれは難しい。この問題については[2]などを参照）。図２に示す内部構造を前提に仕様を記述した。

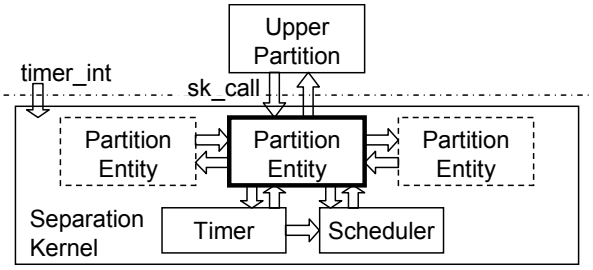

図２　仕様記述のための Separation Kernel のモデル化

(3) XML による仕様記述

主要な構成要素である Partition Entity の仕様を XML 表記拡張状態遷移表として記述した（約 330 行の記述）。その一部を図 3 に示す。記述で使用している変数やメソッドの定義は仕様実行環境の中に含めている。

(4) 自動生成された Java プログラム

XMLSMtoJavaSM により Java プログラム（状態機械を実装した StateMachine メソッド）を自動生成した。図 4 にその一部を示す。

(5) 仕様検証

自動生成された Java の StateMachine メソッドを、仕様実行環境中の PartitionEntity クラスの中に取り込むと、仕様実行が可能になる。ユーザの位置付けにある Upper Partition のスタブを作り、コールを発生させることにより、仕様レベルのテストが行える。さまざまなコール・シーケンスについてテストを実施した。

実行結果の検証方法を次に示す。

・実行ログから自動生成できるシーケンス図（例を図 5 に示す）により、要求シナリオどおりの動作か、の検証。

・状態遷移履歴によりテストの状態遷移の網羅性の確認。

・変数で表されたシステムの状態が、テストの事前と事後で、要求条件に合致していることの検証。

仕様検証の結果、仕様のセマンティクスレベルの抜けなどを見付けることができた。

5. 評価

(1) 拡張状態遷移表を XML 表記し、実行可能な Java メソッドを自動生成することにより、仕様レベル・テストによる仕様検証が可能になった。

(2) XML による記述を変更することにより、仕様を変更したときの検証が容易にできる。

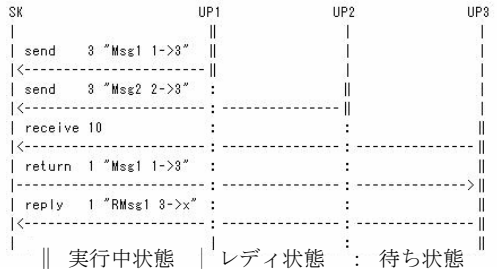

図５　実行ログから生成したシーケンス図の例（部分）

```
<statemachine>
  :
  <input type = "1" name = "send">
    <predicate name = "PREDICATE1">
        :
    </predicate>
    <predicate name = "PREDICATE2">
      Condition = sk.skmode;
      <state number = "2">
        <action>
          if (in.val == pid || ! existPID(in.val)) {
            ret = returnInfo(ERROR);
            state = 5;
            Output(SCHEDULER, SCHEDULE);
          } else {
            timeout = in.val2;
            in.val2 = genID();
            Output(in.val, 6, in.val, in.val2, in.str);
            saveSendInfo(in.val, in.val2);
            state = 3;
          }
        </action>
      </state>
      <stategroup name = "default">
        <action dontcare = "true">
        </action>
      </stategroup>
    </predicate>
  </input>
  <input type = "2" name = "receive">
    :
```

図３　拡張状態遷移表による仕様記述（XML 表記、部分）

```
public boolean StateMachine(IOObj in) {
    :
  switch (in.type) {
    case 1:  // send
        :
        :
      Condition = sk.skmode;
      if (Condition) {
        switch (state) {
          case 2:  // run_in_k
                ここにactionの内容が入る
            break;
          default:
            systemError("The input is not acceptable.");
            return false;
        }
        break;
      }
    case 2:  // receive
      :
```

図４　自動生成された Java メソッド（状態機械、部分）

６．今後の課題

仕様検証手法としての課題を示す。

(1) 形式手法との比較とそれによる改善。

(2) 仕様検証システムとしての汎用化。

付記：研究室 4 年生の笠原良太君と小金憲君に関連ツール（実行ログからシーケンス図作成、XML 表記拡張状態遷移表から通常の表作成）の作成に協力してもらった。

参考文献

[1] 野口 健一郎、元岡 達：オペレーティング・システムの記述に関する一考察、情報処理、Vol. 14, No. 2, pp. 98-105 (1973).

[2] 野口 健一郎：ソフトウェアの論理的設計法、共立出版（1990).

[3] 川守田 慶、野口 健一郎：” 形式手法を用いた組込み用 OS の試作–B メソッドによる仕様検証実験–”, FIT 2008.