2010 年度第 6 回情報処理学会東北支部研究会(山形大学)
資料番号 10-6-B4-4

# モデル検査ツールを用いた Web アプリケーションのテストケース自動生成

金野素良† 本間圭†† 和泉諭††† 阿部雄貴† 富樫敦†† 高橋薫†

†仙台高等専門学校 〒989-3128 仙台市青葉区愛子中央四丁目 16 番 1 号
††宮城大学大学院事業構想学研究科 〒981-3298 宮城県黒川郡大和町学苑 1 番地 1
†††東北大学電気通信研究所/情報科学研究科 〒980-8577 仙台市青葉区片平 2-1-1

**概 要**

本稿では Web アプリケーションの設計で作成される画面遷移図に着目し，アプリケーションの全体の一側面をモデル化する．そのモデルをモデル検査ツールである NuSMV で記述し，テストに必要な Web アプリケーションの成り立つべき検査項目についてのテストケースを生成する．

## １．はじめに

近年のインターネットにおける Web アプリケーションの普及に伴い，オンラインショッピングやオンラインバンキング等，重要なトランザクションを扱うアプリケーションも増加の一途をたどっている．

しかし，Web アプリケーションの設計に焦点を当てると，多くの作業は経験とノウハウに基づいたものであり，設計の自動化はあまり行われていない．また，設計を実装した時にその実装が正しく稼働するかどうかも人手によって行われているのが現状である．特に，Web アプリケーションの設計においては，入力インタフェースである画面単位で設計が行われることが多い．大規模プロジェクトの場合，複数の担当者によって，各画面の設計が行われるため，画面間の連係によりシステム全体が正しく挙動するかは設計段階で確認が難しく，構築後のテストで問題が顕在化するのが大半である．

そこで，Web アプリケーションの設計で作成される画面遷移図に焦点をあて，画面遷移とシステム環境をそれぞれ有限状態オートマトンとして考え，その直積オートマトンを構成することにより，アプリケーション全体の一側面をモデル化する手法が提案されている[5][7][8][9][10]．そこでは，一般的な Web アプリケーションで成り立つべき性質を検査項目として挙げ，モデル検査ツールの SPIN[1][2]を用いることにより検査を行っている．

さらに，信頼性の向上のためには，与えられた Web アプリケーションの設計を実装した時にその実装が正しく稼働するかどうかをテストすることが必要であり，この段階で多くの不具合を発見できることが求められる．

本研究では Web アプリケーションの設計で作成される画面遷移図に着目し，アプリケーション全体の一側面を文献[5]をもとにモデル化する．そのモデルをモデル検査ツールである NuSMV[3][4]で記述し，テストに必要な Web アプリケーションの成り立つべき検査項目についてのテストケースを生成する．

モデル検査では，有限状態システムの全ての状態を網羅的に検査し，検査したい性質を満たしているかどうかを検証式を用いて検証する技術である．また，検査したい性質が満足されていない場

合には，検査項目が満たされていないイベント系列を出力するようになっている．このことを利用して，検証式を否定したもので検査を行えば，アプリケーションをテストするために必要なイベント系列（テストケース）が出力され，そのテストケースを実際のアプリケーションで実行することによって，そのアプリケーションが検査項目を満たしているかどうかがわかる．

以下，本論文では 2 節 でモデル検査について述べ，3 節で Web アプリケーションのモデル化とテストついて述べる．4 節で具体的な適用例を示し，最後にまとめを 5 節で示す．

## ２．モデル検査

モデル検査は，情報処理システムが正しく動作することを検証する方法の１つで，定理証明法とともに数理的技法（形式技法）と呼ばれている．設計段階での適用が可能で，モデル検査を用いることによって設計のバグを早期に発見することができ，開発コストを削減することができる．また，システムを厳密に記述して検証するので，できあがるシステムの信頼性を向上させることができる．一般に， モデル検査では，次のようにして情報処理システムの検証を行う．

1. 情報処理システムを状態遷移系（オートマトン）として記述
2. 情報処理システムに要求する性質（動作仕様，検査項目）を論理式で記述
3. 状態遷移系が論理式を満たすことを示す

定理証明法では，論理的な推論を積み重ねてシステムが仕様を満たすことを示す．これに対してモデル検査では，状態遷移系の動きを計算機の上で模倣（真似）をして，あらゆる場合をしらみつぶしに調べることによって，状態遷移系が論理式を満たすことを示す．

### 2.1　モデル検査とは

有限状態遷移系 S と線形時相論理 LTL（Linear Temporal Logic）の式 p を入力すると，S が p を満たすか満たさないかを出力する効率の良いアルゴリズムが開発されている．「モデル検査」とは，このアルゴリズムを使って有限状態遷移系が LTL の時相論理式を満足するかどうかをコンピュータで自動判断する検証方法である．LTL 以外にも計算木論理 CTL(Computation Tree Logic)などの時相論理を使って同様のことを行う手法もモデル検査という．

モデル検査の大まかな流れを図 2.1 に示す．

1. 検査したい対象をモデル化する．この際に検査したい性質に関して考慮し，その性質を損なわないように行う．
2. 検査したい対象に対して，仕様や満たすべき性質などと言った何を検査したいのかを設定する．この検査項目を時相論理式で表現する．
3. モデル化したものと検査項目を時相論理式で表現した検査式をモデル検査ツールで記述する．この記述はモデル検査ツールの専用言語で記述する．
4. モデル検査を行う．操作は各ツールによる．
5. ツールで行った結果をチェックする．結果は検査式を満たしているか真偽値で出力される．偽を出力した際には反例を出力し，なぜ検査式が偽の判定を行ったかを示す．

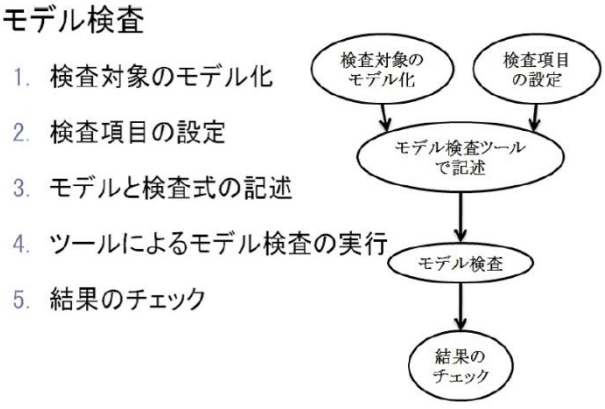


図 2.1　モデル検査の流れ

モデル検査のためのツールがいくつか開発され

ており，中にはオープンソースのシステムもある．代表的なものが「NuSMV」と「SPIN」の二つのシステムであると言える．

NuSMV（New Symbolic Model Verifier）[3][4]はシンボリックな手法を用いた最初のモデル検査器 SMV（Symbolic Model Verifier）を拡張したもので，扱える時相論理式は上で述べたように LTL と CTL の二つである．二分決定グラフ（Binary Decision Diagram, BDD）による手法と SAT ソルバを用いた限定モデル検査手法の両方を用いて検証を行うことができるという特徴をもつ．

SPIN（Simple Promela Interpreter）[1][2] は米国のベル研究所で開発されたツールである．フリーで利用できるツールであり，1991 年に初版が公開されて以来，多くの研究者や技術者に利用されている．SPIN に関する研究開発や SPIN による開発事例を発表する SPIN ワークショップも毎年開催されており，SPIN を利用する際の有力な情報源となっている．また，参考書もいくつか出版されている．現在では Unix 系の OS をはじめとして Windows や Mac でも利用可能である．SPIN では，Promela（Protocol/Process metalanguage）と呼ばれる言語を用いて設計対象システムのモデルを記述し，そのシミュレーションや検証を行う．

## 2.2 NuSMV でのモデル検査

NuSMV では，LTL と CTL の両方を扱えるが，本研究では以下の LTL を用いる．

LTL では変項(素命題) p1，p2，... や通常の論理演算子の他に，以下の時相演算子を使用する：

N （next）
G （globally）
F （in the future）
U （until）

最初の 3 つの演算子は単項演算である．従って，φが論理式であれば，N φ も論理式である．最後の１つは二項演算である．従って，φ と φ が論理式であれば，φ U φ も論理式である．

LTL の論理式は状態遷移系の経路上の逐次的な真理値として評価される．LTL の論理式はその経路上の最初の位置において真であるときのみ真である．演算の意味論は以下のように与えられる．

N φ　：φ は次の状態で真である．（X と表記することもある）
G φ　：φ は今後常に真である．
F φ　：φ は将来のいずれかの時点で真となる．
φ U φ　：φ は現在または将来の時点で真であり，かつ φ はその時点まで真である．その時点以降 φ は真になるとは限らない．

LTL で表現できる重要な特性として次の 2 種類がある．安全性特性は「何か悪いことが決して起こらない」ことを意味する（Gφ）．活性特性は「何か良いことがいずれ起きる」ことを意味する（Fφ）．安全性特性とは，有限な期間での反例を無限の時系列に拡張しても反例であるような状態である．一方活性特性は，有限な期間での反例を無限の時系列に拡張したとき、それが反例でなくなる（その論理式が真となる）状態である．NuSMV による検証の簡単な手順を示す．

1. まずモデルを考え，状態遷移系を考える
2. 次に，その状態遷移系に沿ったソースを作る（プログラムを作る）
3. 検査したい性質を LTL 式にし，検査する

モデル検査の例を図 2.2 に示す．動作仕様は以下となる．

動作仕様
・各変数は同期して値が変化
・変数 A，B は 0 → 1，1 → 2，2 → 0　と繰り返して変化する．ただし B ＝ 1 のときは A は次の時点でもその値を保ち，変化しない。かつ，A=B

のときは次の時点でも保ち，変化しない．初期値は共に 0 である．

・スイッチは，A=B=2 のとき ON，A=B=1 のとき OFF と変化する．それ以外では変化しない．初期値は OFF とする．

図 2.3 はモデル検査を行うために記述したプログラムの一部である．

上記の動作仕様で，以下の性質が満たされているか検査する．

1. スイッチが OFF ならば，いずれ ON になる
2. スイッチが ON ならば，いずれ OFF になる

LTL 式

LTL 式で表すと以下である．

1. G（sw=off → F(sw=on)）
2. G（sw=on → F(sw=off)）

LTL 式で検査式を与えるため，LTLSPEC と記述する．

CTL 式での検査の場合は SPEC と記述する．

図 2.4 がモデル検査を行った時の画面の一部である．

検査項目の 1 つ目は真の値を返しているので，その検査項目の性質を満たしていることを示している．一方，２つ目は偽の値を返している．偽を返しているため，反例が出力され，モデルの状態空間の中で，初期値からどのように変化し，その検査項目を満たしていないかを示している．この反例を見ることによって，修正個所を容易に発見でき，その項目を満たすように修正をしていくことにより，不具合を無くしていくことができる．

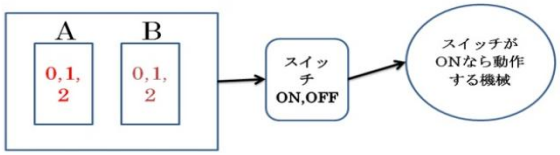


図 2.2　モデル検査の例

```
MODULE main
VAR
        a       :       {0,1,2};
        b       :       {0,1,2};
        sw      :       {on,off};
ASSIGN
        init(a) :=      0;
        next(a) :=      case
                                b=1:    a;
                                1:      (a+1) mod 3;
                                esac;
        init(b) :=      0;
        next(b) :=      case
                                a=b:    b;
                                1:      (b+1) mod 3;
                                esac;
        init(sw) := off;
        next(sw) := case
                                a=2 & b=2: on;
                                a=1 & b=1: off;
                                1:sw;
                        esac;
LTLSPEC G ( sw = off -> F (sw = on ))
LTLSPEC G ( sw = on -> F (sw = off ))
```

図 2.3　モデル検査プログラム

```
コマンド プロンプト
NuSMV terminated by a signal

C:¥Documents and Settings¥Y-Abe¥My Documents¥smv¥NuSMV>NuSMV systemalpha.smv
*** This is NuSMV 2.4.3 (compiled on Tue May 22 14:08:54 UTC 2007)
*** For more information on NuSMV see <http://nusmv.irst.itc.it>
*** or email to <nusmv-users@irst.itc.it>.
*** Please report bugs to <nusmv@irst.itc.it>.

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat
*** Copyright (c) 2003-2005, Niklas Een, Niklas Sorensson

-- specification  G (sw = off ->  F sw = on)  is true
-- specification  G (sw = on ->  F sw = off)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  a = 0
  b = 0
  sw = off
-> Input: 1.2 <-
-> State: 1.2 <-
  a = 1
-> Input: 1.3 <-
-> State: 1.3 <-
  a = 2
  b = 1
-> Input: 1.4 <-
-> State: 1.4 <-
  b = 2
-> Input: 1.5 <-
```

図 2.4　モデル検査の実行

## ３．Web アプリケーションのモデル化とテスト

本節では，Web アプリケーションのモデル化と，その実装時の試験に用いるテストケース生成について述べる．

### 3.1　モデル化

画面遷移図は Web アプリケーションの設計にお

ける重要な要素である．画面遷移図は Web アプリケーションのインタフェースの遷移を規定し挙動の多くを制限するため，アプリケーションの一側面として考えることができる．また，画面遷移図は画面そのものを状態と捉え，画面遷移を状態遷移と見立てることにより状態遷移システムとして考えることが可能である．ただし，画面遷移の特徴として，受理状態は考えないとする．

【定義】画面に着目した状態遷移システム

$M = \langle Q, \Sigma, \delta, q_0 \rangle$

ここで　Q：画面(状態)の有限集合

Σ：入力記号の有限集合

（画面に対するアクション）

δ：動作関係（画面遷移）

（$\delta \subseteq Q \times \Sigma \times Q$）

$q_0$：初期状態（トップ画面）（$\in Q$）

このモデルを変数を用いて拡張する．変数付モデルを導入することにより，変数の値をシステムの状態とせずにマクロ的に定義し，変数の値に依存した状態遷移を動作関係に含めることで，システム全体を見通し良く定義できる．画面から入力される変数の有限集合を $X = \{x_1, \cdots, x_n\}$ とする．また，遷移上にはラベルの他に変数を利用した遷移条件と変数代入を追加すると，状態遷移システムは以下のように定義できる．

【定義】画面に着目した変数付き状態遷移システム

$M = \langle Q, \Sigma, \delta, q_0, \{R_i\}_{i \in [1,n]}, \{v_{0i} | i \in [1,n]\} \rangle$

ここで，　Q：画面（状態）の有限集合

Σ：入力記号の有限集合

（画面に対するアクション）

δ：動作関係（画面遷移）

（$\delta \subseteq Q \times (R_1 \times \cdots \times R_n \rightarrow \{\mathrm{true}, \mathrm{false}\}) \times \Sigma \times Q$）

$q_0$：初期状態（トップ画面）（$\in Q$）

$R_i$：画面から入力された変数 $x_i$ の有限な変域

$v_{0i}$：変数 $x_i$ が表示される画面における初期値（$\in R_i$）

上記定義において，動作関係の第 2 要素は，変数の値による遷移条件を表す．

### 3.2　NuSMV を用いたテストケース生成

上記の Web アプリケーションモデルを用いて，テストケースの生成方法を示す．定義したモデルは，設計段階で作成される．したがって，システムテストの段階で行われるテストケースはアプリケーションの設計と一致する．システムテストのために生成されたテストケースは，機能的な構造のテストであるブラックボックステストに適用される．

Web アプリケーションのモデルと合わせて考えると，各状態とイベントをテストのターゲットとみなすことができる．したがって，テストケースは．以下のパターンの組み合わせである．

(1) 状態(画面)

(2) 発生するイベント

テストケースは次の LTL 式を使用し，画面 p から画面 q への遷移を表すイベント系列を生成する．これは，NuSMV によって出力された反例を利用することで行うことができる．

F( p & F(q) )

ここで，Fは将来を表す演算子であり，＆は論理積を表す．

上記 LTL 式の否定をとり，NuSMV で検査を実行すると，遷移可能な場合にエラーとなり，その反例として画面 p と画面 q との間のルートが出力され，それをテストケースとして用いる．

具体的な例として，テストの状態としてある画面の状態遷移システムＡの画面 X と画面 Z がある

と仮定する．そして，画面 X を p に，画面 Z を q にセットし，モデルを NuSMV で記述し，検査を行う．F( p & F(q))が満たされない，すなわち X から Z へ到達可能である場合，X から Z に到達するパスが反例から得られる．そのパスに含まれるイベントの配列は，X から Z に到達可能な状態のテストケースとして使用することができる．

## 4．適用例

本節では，テストするWebアプリケーションの例として，e-shoppingアプリケーションを対象とする．実際にアプリケーションを作成し，上記の状態遷移システムに則り，モデル化を行い，テストケースを生成し，テストを行う．

### 4.1 e-shopping アプリケーションの設計と実装

Amazon.com[14]を参考に，図 4.1 のような画面遷移をもつショッピングサイトを設計し，実装した．

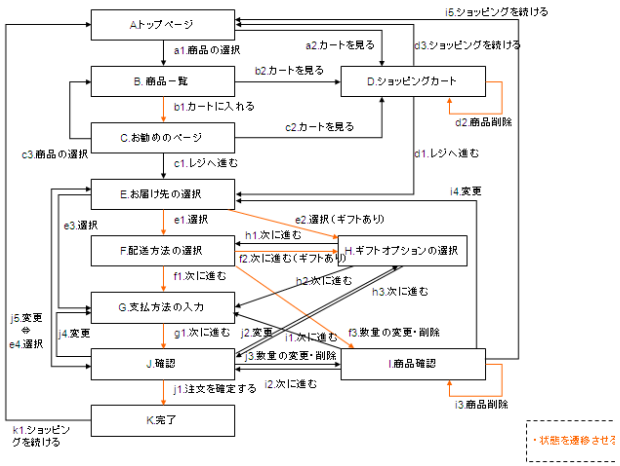


図 4.1　e-shopping アプリケーションの画面設計

使用した言語は Java と HTML で，Web サーバでページを動的にするため，jsp とサーブレットで構成した．ツールはプログラムの編集作成に eclipse，jsp とサーブレットを動作させるため Tomcat5.5，データベースには MySQL を使用した．

### 4.2 テストケース生成とテスト実施

画面遷移のモデルを図 4.2 に示す．

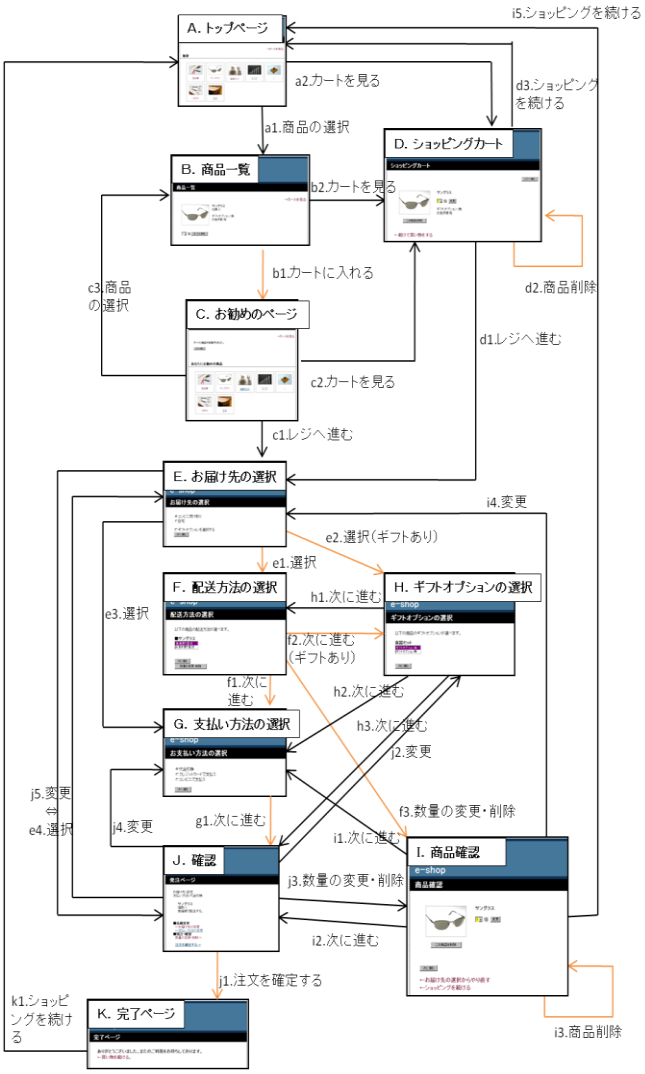


図 4.2　画面遷移モデル

なお，このモデルにおいて，画面から次画面に移る際はイベントによる遷移として表現されるが，NuSMV では，状態遷移におけるイベントの記述はできない．そのため NuSMV で表現するために，図 4.3 に示すように記述する．

図 4.3 では，画面と，その画面に至るイベントを 1 つの状態として考える．この例では，画面 x にいたるイベントを ev0，画面 x から画面 y にいたるイベント ev1 とした場合を示している．ただし，初期状態での画面ではイベントはないものとし，0 と表現する．

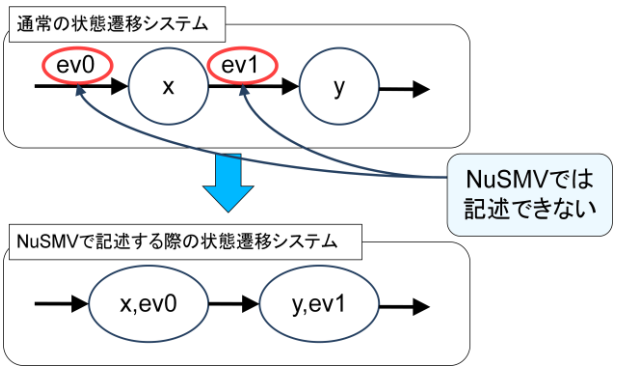


図 4.3 NuSMV でのモデル化

以上の方法でモデル全体を SMV コードで記述した一部を以下に示す.

```
MODULE main
VAR

screen                  :
      {a,b,c,d,e,f,g,h,i,j,k};

event          :
      {0,a1,a2,b1,b2,c1,c2,c3,
       d1,d2,d3,e1,e2,e3,e4,
   f1,f2,f3,g1,h1,h2,h3,i1,
   i2,i3,i4,i5,j1,j2,j3,j4,j5,k1
 };

giftoption   :        {0,1};
----giftoption
p            :        {0,1};
----gift gamen event
q            :        {0,1};
----shiharai gamen event
r            :        {0,1};
----kakunin gamen event

item         :        {0,1};
----item
                  :
```

以下，実際にテストケースを生成する例を示す．

図 4.2 より，テストケースを出す対象の画面を「A．トップページ」と「K．完了画面」とし，3 章で記述した LTL 式 F( p & F(q) )の p と q にそれぞれあてはめる．この LTL 式を用いて gNuSMV で検査すると図 4.4 のように反例が出力される．

LTL property 0: ! F (screen = a & F screen = k)

| Loop | Step | event | giftoption | item | p | q | r | screen |
|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | a |
| | 1 | a1 | 0 | 0 | 0 | 0 | 0 | b |
| | 2 | b1 | 0 | 1 | 0 | 0 | 0 | c |
| | 3 | c1 | 0 | 1 | 0 | 0 | 0 | e |
| | 4 | e3 | 0 | 1 | 0 | 0 | 0 | g |
| | 5 | g1 | 0 | 1 | 0 | 1 | 0 | j |
| | 6 | j1 | 0 | 1 | 0 | 1 | 1 | k |
| | 7 | k1 | 0 | 0 | 0 | 1 | 1 | a |
| | 8 | a1 | 0 | 0 | 0 | 0 | 0 | b |
| | 9 | b1 | 0 | 1 | 0 | 0 | 0 | c |
| | 10 | c1 | 0 | 1 | 0 | 0 | 0 | e |
| | 11 | e3 | 0 | 1 | 0 | 0 | 0 | g |
| | 12 | g1 | 0 | 1 | 0 | 1 | 0 | j |
| | 13 | j1 | 0 | 1 | 0 | 1 | 1 | k |
| | 14 | k1 | 0 | 0 | 0 | 1 | 1 | a |

図 4.4　反例の出力

図 4.4 において，四角で囲った部分が「A．トップページ」から「K．完了画面」へのイベント系列であり，テストケースとしては

**a1.商品の選択 → b1.カートに入れる → c1.レジへ進む → e3.選択 → g1.次に進む → j1.注文を完了する**

となる．

他の画面間の遷移についても同様に生成し，前節の e-shopping サイトの実装に適用する(図 4.5)．そうして得られた結果から，テストの合否を判断する．

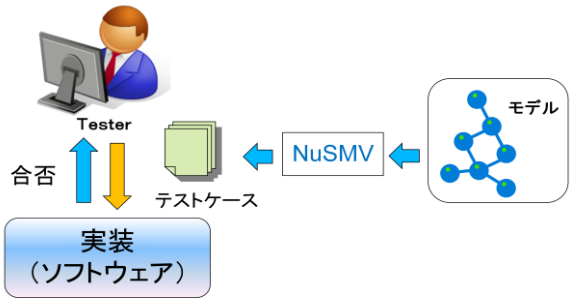


図 4.5　テストのイメージ

結果，全てのテストに合格し，実装が仕様通りに動作していることを確認できた．

また，実装でわざと遷移を間違えたものに対し

てテストしたときは，目的の画面へと遷移しないことがわかり，設計と実装の不一致が確認できた．

## ５．まとめ

本研究では，Web アプリケーションの設計において重要な要素である画面遷移図を，状態遷移として考え，モデル検査の手法を用いてモデル化を行った．また，そのモデルと検証式を NuSMV で記述し，Web アプリケーションをテストするためのテストケースを生成した．

具体的には，検査したい性質を LTL 式で記述し，それをモデル検査ツール NuSMV で網羅的に検証し，反例を出力させ，その反例のイベントを表す変数の値をテストケースとして使用した．

また，テストケースを実際の Web アプリケーションに適用するため，e-shopping アプリケーションの作成を行った．このアプリケーションに生成したテストケースを適用し，画面設計と実装が仕様通りであるかどうかをテストすることができた．

これにより，与えられた Web アプリケーションの設計を実装した時に，その実装が正しく稼働しているかどうかをテストする際，使用するテストケースの生成が容易になったと考えられる．

今後は，画面遷移図だけに限らず，内部の変数や状態にも着目し，より信頼性の高めるためのテストケース生成を考えていく必要がある．

## 参考文献


[1] http://spinroot.com/

[2] G. J. Holzmann, “The Model Checker SPIN,” IEEE Transactions on Software Engineering, Vol. 23, No. 5, pp. 279-295, 1997.

[3] A. Cimatti et al., “NuSMV: A New Symbolic Model Verifier,” LNCS Vol.1633, pp.495-499, 1999.

[4] “NuSMV,” http://nusmv.irst.itc.it/

[5] K. Homma, S. Izumi, K. Takahashi and A. Togashi, "Modeling and Verification of Web Applications Using Formal Approach," International Journal of Computer and Information Science (IJCIS), Vol.11, No.2, 2010.

[6] 門野雅弥, “モデル検査器 NuSMV を利用したテストケース自動生成，” 2008-SLDM-134/2008-EMB-8 (27), 2008.

[7] 佐藤祐太，本間圭，高橋薫，和泉諭，阿部雄貴，富樫敦，"形式的手法と検査ツールによるモデル検査事例と考察," 情報処理学会第 72 回全国大会, 2010.

[8] 佐藤祐太，本間圭，鈴木博勝，和泉諭，高橋薫，富樫敦，"形式的手法と検査ツールに基づく Web システムの設計と検証," 2009 年度第 5 回情報処理学会東北支部研究会，A-3-1, 2010.

[9] 本間圭，高橋薫，和泉諭，阿部雄貴，富樫敦，"変数を用いた Web アプリケーションのモデル化と形式的手法による検査," 電子情報通信学会ソフトウェアインタプライズモデリング研究会，信学技報，Vol.109, No.298, SWIM2009-17, pp.31-38, 2009.

[10] 本間圭，高橋薫，富樫敦，"形式的手法による Web アプリケーションのモデル化と検証," 電子情報通信学会ソフトウェアサイエンス研究会，信学技報，Vol.109, No.40, SS2009-8, pp.43-48, 2009.

[11] 産業技術総合研究所システム検証センター，” 4 日で学ぶモデル検査 初級編，” NTS, 2006.

[12] 田中譲，“ソフトウェア科学基礎，” 近代科学社, 2008.

[13] 阿部雄貴，” モデル検査を用いたオントロジー検証，”仙台高専平成 21 年度卒業研究論文, 2010.

[14] http://www.amazon.com/