画面遷移仕様のモデル検査

崔 銀惠　　河本 貴則　　渡邊 宏

産業技術総合研究所　システム検証研究センター

画面遷移仕様のモデル検査*

崔 銀惠　河本 貴則　渡邊 宏
Eun-Hye CHOI　Takanori KAWAMOTO　Hiroshi Watanabe
産業技術総合研究所 システム検証研究センター
〒661–0974 尼崎市若王寺 3–11–46
E-mail: {e.choi, t-kawamoto, hiroshi-watanabe}@aist.go.jp

概 要

本研究では，Web アプリケーションなどの設計において必要不可欠である画面遷移に関する仕様を形式的に検証する手法を提案する．提案法では，画面遷移に関する仕様と画面遷移と連携したプログラム処理の流れを記述したフローチャート間の整合性をモデル検査法を用いて検証する．モデル検査法は，システムをモデル化した状態遷移系がシステムの性質を表現した時相論理式を満たすか否かを形式的に検査する手法である．提案法では，Web アプリケーションの画面遷移図とプログラム処理を記述したフローチャートから Kripke モデルを作成し，画面遷移の仕様を命題時相論理 CTL で表現して，モデル検査を適用する．更に，提案法の有効性を確認するため，実際に存在する Web アプリケーションの設計仕様に対して適用実験を行った．実験の結果，デッドロックを含む幾つかの不具合を提案法を用いて短時間で発見できた．

Abstract

In this paper, we propose a formal method for verifying a requirement of page transitions, which is one of the most essential requirements especially for Web-based applications. The proposed method checks whether a page transition requirement is satisfied in a state transition specification of applications, such as a flowchart or an UML activity diagram, based on model checking. We applied the proposed method to real specifications of a Web application used in a certain company and found several errors including deadlock of the specifications in a short time.

1　はじめに

近年，Web アプリケーションの需要は急速に増加しており，電子商取引に代表されるインターネット上のアプリケーションだけでなく企業イントラネット上の業務管理アプリケーションなどを含む様々なシステムが Web ベースで構築される傾向にある．これに従って，Web アプリケーションの信頼性を保証する検証技術の重要性もますます高まってきている．しかし，Web アプリケーションの設計支援・分析・テストを行う技術が数多く存在するのに対して，形式的に検証を行う手法はほとんど提案されていない[1]．本研究の目的は，Web アプリケーションの設計支援のための形式的検証法を提案すること，及び，実際に設計されたアプリケーションに対して提案法を適用し有効性を評価することである．

一般に Web ベースの業務系アプリケーションの設計では，業務担当者などの発注者とソフトウェア技術者からなる受注者の異なる二者が携わることが多い．発注者側は，業務に必要な処理データ内容や画面遷移仕様などを最も基本的な要求仕様として作成し発注を行う．一方，受注者側は，発注者側の仕様をもとに入出力データの処理，ユーザ認証，イベント管理といった詳細なプログラム処理の流れ (フローチャート) を基本的な設計仕様として作成する．この要求仕様とフローチャートの間で食い違いがあると，発注者側の

*本研究は，文部科学省科学技術振興調整費（若手任期付支援），関西電力株式会社との共同研究の研究成果である．
[1]Web サービスやアプリケーションを対象とした形式的検証法に関する国際会議 (WWV'05[11]) がようやく始まろうとしている．

要求が受注者側のプログラマーへ正しく伝わらず，発注者の要求通りのシステムが実現されない．従って良いシステムを作るためには，発注者と受注者の間で取り扱うフローチャートが要求仕様を満たしていなければならない．

フローチャートが要求仕様を満たすことは発注者と受注者の両者が査読をして確認するが，フローチャートと要求仕様との間の食い違いを見ぬけないことが多い．その理由として，次のようなものが考えられる．第一に，システムが複雑になると処理の種類が多様になりフローチャート上の実行系列が増えるために見落としが生じる．第二に，発注者は業務の専門家ではあるがソフトウェア技術者ではないことが多いため，フローチャート上の複雑な流れをすべて理解することは難しい．第三に，受注者はフローチャートを書く側であるため，思い込みを排除して査読をすることが難しい．そこで，フローチャートが要求仕様を満たしていることを発注者側でも機械的に検査できる手法があれば便利である．今回は，Web アプリケーションの要求仕様にある処理データ内容は無視して，画面遷移図とプログラム処理を記述したフローチャート上の画面遷移関係の間の整合性に着目する．画面遷移とフローチャート間の整合性を形式的に検証する手段として，モデル検査法を用いる．

モデル検査 [1] とは，有限状態遷移系でモデル化されたシステムが時相論理で表現した性質を満たすか否かをシステムが取り得る全状態空間を探索して網羅的に検査する手法である．検査したいシステムの状態遷移系のモデルと時相論理式の検査項目が決まれば，既存のモデル検査ツールを利用してある程度の大きさのシステムに対しては全自動的に検査できるという長所があり，近年最も注目されているシステム検証技術の 1 つである．また，モデル検査の結果，検査項目が正しくない場合は反例が出力されるので不具合部分を特定するためにも有効な手法である．

本研究では，まず，どのような場合にフローチャートが画面遷移仕様を満たすのかを考察し，フローチャートが画面遷移仕様を満たすことの数理的な定義を求める．次に，その数理的定義と，フローチャートに画面情報を付けた Kripke モデルの上で CTL(Computation Tree Logic) のある論理式が成り立つことが等価であることを示す．これを利用して，画面遷移に関する整合性の検査法として，画面情報を付けた Kripke モデル上で CTL 式をモデル検査することを提案する．

さらに，提案法の有効性を確認するため，実際に動作している Web ベース業務処理アプリケーションに対して適用実験を行った．実験の結果，与えられた画面遷移図とフローチャートが整合しない遷移関係がデッドロックを含めて幾つか発見できた．また，今回の作業は発注者でも受注者でもない第三者が短時間で行えた．

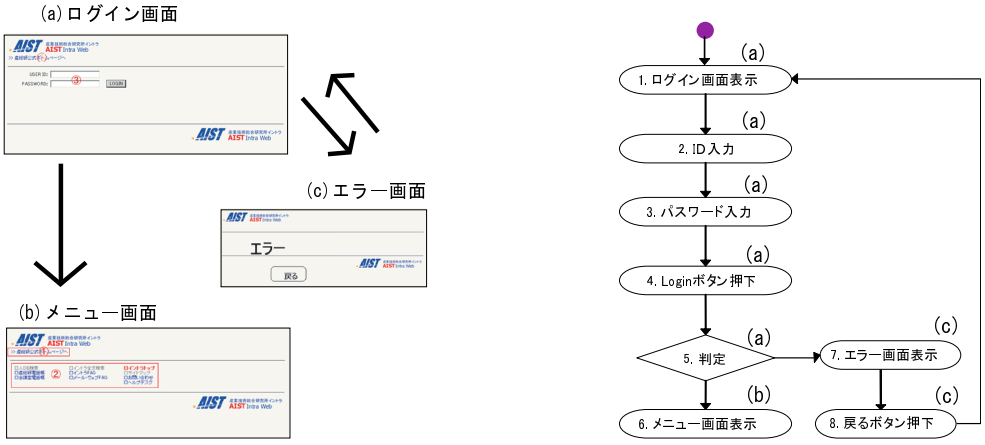
2 関連研究

我々の知る限り，Web アプリケーションを対象とした形式的検証法はまだほとんど提案されていない．

文献 [7] では，Web アプリケーションの画面遷移関係などを UML[9] としてモデル化し，そのモデルを視覚化するツールが提案されている．提案されたツール上ではモデルの分析およびテストが可能であるが，本研究のような形式的な検証は考慮していない．

一方，文献 [4] は，Web アプリケーションの画面遷移関係を含む振舞いを Web オートマトンと呼ばれるオートマトンとしてモデル化し，Web オートマトンをテストに用いる手法を提案している．Web オートマトンは Web アプリケーションの MVC モデル [9] に基づいて作成され，その振舞いからテストパターンが作成される．しかし，形式的検証は今後の課題となっている．

また，文献 [8] では，Web アプリケーションの画面遷移をグラフモデルとして作成しモデル検査を行う形式的手法について述べている．しかし，文献 [8] と本研究の提案法では注目する検証対象と性質が異なる．文献 [8] では Web アプリケーションの保守性に注目して画面遷移が満たすべき好ましい性質を CTL 式で記述し検証するフレームワークが提案されている．



(1) 画面遷移図　　(2) 画面付きフローチャート

図 1: Web 画面遷移図とフローチャートの例

3　提案法

3.1　画面遷移仕様とフローチャート

本論文での**画面遷移仕様**とは，画面の集合 X，画面の間の遷移の集合 $R \subseteq X \times X$ から成る状態遷移グラフ $V = (X, R)$ のことである．画面遷移仕様 $V = (X, R)$ に対して，V **の画面付きフローチャート**とは，状態遷移グラフ (S, T) と写像 $m : S \to X$ からなる三つ組 $(S, T, m : S \to X)$ のことである．ここで，状態遷移グラフ (S, T) はフローチャートのノードと遷移関係を表現し，写像 m はフローチャートの各ノードに対応する画面を割り当てる．

例 1　業務処理システムの設計では，最初に画面遷移図を要求仕様として作成し，次に，画面遷移図の各画面ごとにその中での処理を手続き化してフローチャートを作成する．したがって，画面遷移仕様を詳細化したフローチャートの各ノードに一つ一つ画面を割り当てることが可能である．そこで，このようなフローチャートは画面付きフローチャートでモデル化できる．画面付きフローチャートを図 1 の (2) のように，フローチャートの各ノードの上に対応する画面を書いて表現する．　□

3.2　整合性の定義

ここでは，画面付きフローチャートが画面遷移仕様を満たすことの定義を与える．まず第一に，画面遷移仕様にある画面遷移一つ一つについて，それを実現する遷移が画面付きフローチャートの中のどこかになければ，画面付きフローチャートが画面遷移仕様を実現しているとは言えない．一方，フローチャートのノードの間の遷移が起こったときに，画面遷移仕様に従い画面の遷移が起こる，あるいは画面は変化しないはずである．これらをまとめて，次のように定義する．

定義 1　画面遷移図 $V = (X, R)$ と V の画面の付いたフローチャート $D = (S, T, m : S \to X)$ が与えられているとする．　次の二つの条件が満たされるとき，**画面付きフローチャート** D **が画面遷移仕様** V **を満たす**という：

(C1) V の各遷移 $(x_1, x_2) \in R$ に対し，$m(s_1) = x_1$ かつ $m(s_2) = x_2$ となる遷移 $(s_1, s_2) \in T$ が D に存在する，

(C2) D の各遷移 $(s_1, s_2) \in T$ に対して，$m(s_1) = m(s_2)$ または $(m(s_1), m(s_2)) \in R$ が成り立つ．　□

条件 (C1) は画面遷移に対応するフローチャート上の遷移があることを，条件 (C2) はフローチャートの遷移に対応する画面遷移があることを意味する．以降，画面付きフローチャートが条件 (C1) と (C2) を満たすとき，すなわち，画面付きフローチャートが画面遷移仕様を満たすとき，画面遷移図とフローチャートの間で整合が取れているという．

3.3 検査項目

ここでは，画面付きフローチャートが画面遷移仕様を満たすことと等価なモデル検査問題を挙げる．

命題時相論理 CTL は命題論理に，時間に関する 6 個の単項論理演算子 AG, EG, AF, EF, AX, EX と 2 個の二項論理演算子 AU, EU を加えて拡張したものである．ここでは，本研究で用いる演算子 AX, EX, AG, EF のみを紹介する．(CTL に関するより詳しい説明は [3] を参考にされたい.)

CTL の論理式は Kripke モデルの各状態の上で，CTL 構文の定義に従い，帰納的に解釈される．Kripke モデルは状態遷移グラフとその各状態に原始命題の集合を割り当てる関数から成る．Kripke モデルとその上の状態（ここでは s とする）を 1 つ固定したとき，状態 s の上で CTL 論理式の意味は次のように与えられる：

- $AX\ \phi$: s の次の全ての状態で ϕ が成り立つ.
- $EX\ \phi$: ϕ が成り立つ s の次の状態がある.
- $AG\ \phi$: s から到達可能な全ての状態で ϕ が成り立つ.
- $EF\ \phi$: s から ϕ が成り立つ状態へ到達可能である.

ここで，Kripke 構造の遷移関係は離散的な時間変化に対応する状態の変化を表しており，s から直接遷移可能な状態を「s の次の状態」と呼ぶ．また，到達可能関係は遷移関係の反射的推移的閉包である．従って，s から到達可能な状態は s 自身も含み，s から s' へ到達可能かつ s' から s'' へ到達可能であれば s から s'' へ到達可能である．Kripke モデル K の状態 s で CTL 論理式 ψ が成り立つことを記号 $K, s \models \psi$ と記述する.

定理 1 画面遷移図 $V = (X, R)$ と，その画面付きフローチャート $D = (S, T, m)$ が与えられていて，D には，すべての状態へ到達可能な特別な状態 $s_0 \in S$ があることを仮定する．このとき，D が画面遷移仕様を満たすことと，D から決まる原始命題集合 X 上の Kripke モデル $K_D = (S, T, L_m : S \to \mathcal{P}(X))$ が次の二種類の CTL 論理式から構成される条件を満たすことは等価である：

(e1) 各 $(x_1, x_2) \in R$ に対して，$K_D, s_0 \models EF(x_1 \wedge EX\ x_2)$,

(e2) 各 $x \in X$ に対して，$K_D, s_0 \models AG(x \to AX(x \vee \bigvee_{(x,x') \in R} x'))$.

ただし，K_D のラベル関数 $L_m : S \to \mathcal{P}(X)$ は $L_m(s) = \{m(s)\}$ で定義されるものとする． □

(e1) と (e2) の論理式の意味を考えるとそれぞれ次の 1, 2 となり，(C1), (C2) と等価である.

1. V 上の各画面遷移 (x_1, x_2) に対して，Kripke モデル K_D 上に x_1，x_2 が連続して成り立つ状態がある.
2. Kripke モデル K_D 上のある状態で x が成り立つ場合，次の状態では再び x が成り立つか，あるいは，$(x, x') \in R$ となる x' が成り立つ.

定理 1 で述べた同値性を使い，画面付きフローチャートが画面遷移仕様を満たすことをモデル検査の問題として検査することができる．一般的に，画面付きフローチャートの任意の状態はフローチャートの初期状態から到達可能であることが想定されるので，定理 1 の仮定は妥当であろう．本研究ではこの仮定の下，

画面付きフローチャートの初期状態を定理 1 の特別な状態 s_0 とし，その初期状態で (e1) と (e2) の CTL 式が成り立つかをモデル検査する．さらに，デッドロックや到達性といった性質も CTL で記述してモデル検査できる．これについては，4 節で述べる．

3.4 模擬実験

ここでは，図 1 の画面遷移図と画面付きフローチャートを具体例に提案法を説明する．

3.4.1 Kripke モデル

図 1-(2) の画面付きフローチャートから次の Kripke モデル K_e が決まる：

$$
\begin{array}{lcl}
K_e & = & (S_e, T_e, L_e : S_e \to \mathcal{P}(\{a,b,c\}), \\
S_e & = & \{1,2,3,4,5,6,7,8\}, \\
T_e & = & \{(1,2),(2,3),(3,4),(4,5),(5,6),(5,7),(7,8),(8,1)\}, \\
L_e(s) & = & \{a\} \text{ if } s \in \{1,2,3,4,5\}, \\
 & & \{b\} \text{ if } s \in \{6\}, \\
 & & \{c\} \text{ if } s \in \{7,8\}.
\end{array}
$$

この Kripke モデル K_e では，状態 $1(\in S_e)$ から任意の状態へ到達可能である．この状態 1 を定理 1 の特別な状態 s_0 とする．

3.4.2 検査項目

定理 1 より，図 1 の画面付きフローチャートが画面遷移仕様を満たすことと，上の 3.4.1 の Kripke モデル K_e の状態 s_0 で次の 6 つの CTL 式が成り立つことは等価である．

1. $EF(a \wedge EX\ b)$
2. $EF(a \wedge EX\ c)$
3. $EF(c \wedge EX\ a)$
4. $AG(a \to AX(a \vee b \vee c))$
5. $AG(b \to AX(b))$
6. $AG(c \to AX(c \vee a))$

3.4.3 モデル検査

さて，実際に 3.4.1 の Kripke モデルで 3.4.2 の論理式が成り立つかをモデル検査する．

本研究では，記号モデル検査 (SMV, Symbolic Model Checking)[5] のツールである NuSMV[6] を用いて検査を行った．SMV はチャネルを使った同期通信などの複雑なモデルを記述しにくい反面，SPIN[2] や UPPAAL[10] といった他のモデル検査法と比較して状態数の多いモデルを扱うことができ，高速であるという特徴を持つ．

```
 1  MODULE main
 2
 3   VAR
 4    s : 1..8;
 5    x : {a,b,c};
 6    fg : boolean;
 7
 8   ASSIGN
 9    init(s) := 1;
10    init(x) := a;
11    init(fg) := 0;
12    next(s) := case
13                 s=1:         2;
14                 s=2:         3;
15                 s=3:         4;
16                 s=4:         5;
17                 s=5 & fg:    6;
18                 s=5 & !fg:   7;
19                 s=7:         8;
20                 s=8:         1;
21                 1:           s;
22               esac;
23    next(x) := case
24                 s=5 & fg:    b;
25                 s=5 & !fg:   c;
26                 s=8:         a;
27                 1:           x;
28               esac;
29    next(fg) := {0,1};
30
31  SPEC EF (x=a & EX x=b)
32  SPEC EF (x=a & EX x=c)
33  SPEC EF (x=c & EX x=a)
34  SPEC AG (x=a -> AX (x=a | x=b | x=c))
35  SPEC AG (x=b -> AX (x=b))
36  SPEC AG (x=c -> AX (x=c | x=a))
```

図 2: SMV プログラム

提案法では，画面情報付きフローチャートの Kripke モデル K と (e1), (e2) に基づき作成する検査項目を SMV 言語で記述し，NuSMV を用いて検査する．検査結果が真である場合は画面遷移図とフローチャート間で整合が取れていることを確認でき，検査結果が偽の場合は出力される反例を解析して不具合部分を特定する．

実行例 1　図 2 は，3.4.1 節の Kripke モデル K_e と 3.4.2 節の検査項目を SMV 言語で記述したものである．プログラムの 3–6 行目の変数宣言部では，K_e の状態を表すための変数 `s`，画面を指し示す変数 `x`，フローチャートの条件分岐先を非決定的に決めるための変数 `fg` が宣言されている．8–29 行目の状態遷移系記述部では，各変数の初期値と次の状態で取る値を記述している．たとえば，9 行目の $init(s) := 1$; は K_e の初期状態が 1 であると指定している．31–36 行目は 3.4.2 の 1 から 6 の CTL 式に対応する検査項目を記述している．SMV では K_e の初期状態 1 において SPEC の後に記述した式が成り立つか否かが検査される．

図 2 のプログラムを NuSMV ツールで実行すると，実行結果はすべての検査項目に対して真となる．従って，図 1 の画面付きフローチャートは画面遷移仕様を満たすことが確認できる．

実行例 2　次に，反例が出る検査例を見せるために，図 1-(2) のフローチャートから遷移 (8,1) を仮に除いてみたとしよう．その Kripke モデルは $K'_e = (S_e, T_e - \{(8,1)\}, L_e)$ で与えられ，対応する SMV 言語の記述は図 2 のプログラムからちょうど 20 行と 26 行を削除したものになる．このプログラムを NuSMV に入力して検査した結果が図 3-(1) である．実行結果は検査項目 $EF\ (x = c\ \wedge EX\ x = a)$ が K'_e 上で成り立たないこと示している．すなわち，画面遷移図にある画面 (c) から (a) への遷移に対応する遷移がフローチャート上には無いことが検出された．

実行例 3　今度は，図 1-(2) に遷移 (8,6) を加えたフローチャートを考えよう．その Kripke モデルは $K''_e = (S_e, T_e \cup \{(8,6)\}, L_e)$ で与えられる．K''_e と検査項目を SMV 言語で記述したプログラムを NuSMV に入力

```
-- specification EF (x=a & EX x=b) is true
-- specification EF (x=a & EX x=c) is true
-- specification EF (x=c & EX x=a) is false
-- as demonstrated by the following execution
 sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  fg = 0
  x = a
  s = 1
-- specification AG (x=a -> AX ((x=a | x=b)
 | x=c)) is true
-- specification AG (x=b -> AX x=b) is true
-- specification AG (x=c -> AX (x=c | x=a))
 is true
```

(1) 実行例 1

```
(前略)
-- specification AG (x=c -> AX (x=c | x=a))
 is false
-- as demonstrated by the following execution
 sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 2.1 <-
  fg = 0
  x = a
  s = 1
-> State: 2.2 <-
  s = 2
-> State: 2.3 <-
  s = 3
-> State: 2.4 <-
  s = 4
-> State: 2.5 <-
  s = 5
-> State: 2.6 <-
  x = c
  s = 7
-> State: 2.7 <-
  s = 8
-> State: 2.8 <-
  x = b
  s = 6
```

(2) 実行例 2

図 3: NuSMV の実行結果例

して検査した結果が図 3-(2) である．実行結果は検査項目 $AG(x = c \rightarrow AX(x = c \vee x = a))$ が成り立たないことを示している．すなわち，画面 (c) から (a) 以外の他の画面への遷移がフローチャート上にあることが検出された．出力された反例から，画面遷移仕様にはない，(c) から (b) への遷移があることが読みとれる．

4 適用実験

4.1 実験内容

提案法の有効性を評価するため，実際にある企業で稼働中の Web ベースの事務処理アプリケーションの設計仕様に対して提案法の適用実験を行った．実験では，当該 Web アプリケーションの数十個のモジュールの内の二つ（ここでは，M1 と M2 と呼ぶ）の設計仕様を対象とした．当核 Web アプリケーションの設計仕様は上位と下位の仕様に分かれており，上位の設計仕様は企業内の業務部門が作成し，下位の設計仕様は企業内のシステム部門が作成した．上位と下位の設計仕様にはそれぞれ数種類の仕様が含まれており，上位の仕様には画面遷移図が，下位の仕様には画面遷移図を詳細化した UML の活動図 (Activity Diagram) が含まれていた．

実験ではまず，M1，M2 それぞれに対して，UML 活動図の各状態に画面情報を付けることで画面付きフローチャートの Kripke モデルを作成し，画面遷移図から定理 1 に従った検査項目を作成した．次に，作成したモデルと検査項目を SMV 言語で記述し，モデル検査ツール NuSMV に入力して検査した．

表 1 は，M1，M2 それぞれの仕様の大きさと実験結果をまとめている．表 1-(1) は，与えられた画面遷移図と UML 活動図の大きさ，作成した SMV プログラムの行数と NuSMV を用いたモデル検査に必要だった実行時間を表す．実行時間は全検査項目を検査するために必要な時間の合計であり，M1，M2 のどちらの場合も短時間であった．なお，画面遷移図と UML 活動図から Kripke モデルと検査項目を作成し SMV プログラムとして記述するには，M1，M2 それぞれに対して仕様の作成者でない第三者一人で約 3 時間かかった．

表 1: 実験対象と実行結果

(1) 仕様の大きさと実行時間

		M1	M2
画面遷移図	状態数	9	9
	遷移数	11	11
UML 活動図	状態数	66	56
	遷移数	83	73
SMV プログラム	行数	103	105
	実行時間	0.010s	0.010s

(2) 検査結果と不具合数

	M1			M2		
	(e1)	(e2)	合計	(e1)	(e2)	合計
検査項目数	11	9	20	11	9	20
偽判定項目数	1	5	6	1	4	5
不具合数	1	9	10	1	8	9

4.2 実験結果と考察

表 1-(2) は，定理 1 の (e1)，(e2) それぞれに関する検査項目数と実験結果をまとめたものである．偽判定項目数は検査結果が偽であった検査項目の数を，不具合数は検査結果と反例から検出した整合しない画面遷移の数を表す.

モジュール M1 と M2 に対する検査項目 20 個の内，(e1) に関する検査項目が 11 個，(e2) に関する検査項目が 9 個である．(e1) の検査項目は，画面遷移に対応する UML 活動図の遷移があることを検査する項目で，検査項目数は画面遷移図の遷移数に等しい．一方，(e2) の検査項目は UML 活動図の遷移に対応する画面遷移があることを検査する項目で，検査項目数は画面遷移図の状態数に等しい．実験結果，M1 に関しては，(e1) の検査項目 11 個の内の 1 つ，(e2) の検査項目 9 個の内の 5 つが偽判定であった．また，M2 に関しては，(e1) の検査項目 11 個の内 1 つ，(e2) の検査項目 9 個の内 4 つが偽判定であった.

偽判定の検査項目についてはその反例を解析し，設計仕様における画面遷移の不具合が M1 に関しては 10 個，M2 に関しては 9 個見つかった．M1 に関する画面遷移の不具合の内，(e1) に関する偽判定検査項目から検出できた 1 個の不具合は，画面遷移に対応する UML 活動図の遷移がなかったものある．また，(e2) に関する偽判定の検査項目とその反例から検出できた 9 個の不具合は，UML 活動図の遷移に対応する画面遷移がなかったものある．一方，M2 の仕様に関しても，画面遷移に対応する UML 活動図の遷移がなかった不具合が 1 個，UML 活動図の遷移に対応する画面遷移がなかった不具合が 8 個検出された.

モジュール M1 と M2 に対して発見した合計 19 個の画面遷移に関する不具合の性質は次の 4 つに分けられた.

場合 1 画面遷移図ではある画面 a は画面 b または画面 c へ直接遷移するのに対して UML 活動図では画面 a は画面 b を通してのみ画面 c へ遷移できる.（これに関する不具合遷移数：3）

場合 2 画面遷移図では画面 a は画面 b へのみ直接遷移するのに対して UML 活動図では画面 a は画面 b の他に画面 c へも直接遷移できる.（これに関わる不具合遷移数：1）

場合 3 UML 活動図には画面 a からエラー画面への遷移があるのに対して，画面遷移図にはエラーへの遷移がない.（これに関する不具合遷移数：3）

場合 4 UML 活動図には画面 a から戻り先画面（戻るボタンが押されたときに移る画面）への遷移があるのに対して，画面遷移図には画面 a から戻り先画面への遷移がない.（これに関する不具合遷移数：12）

上記の 4 つの場合の内，場合 1 と場合 2 は上位仕様である画面遷移図と下位仕様である UML 活動図との間で画面の遷移先が違うという問題を引き起こしている．この種類の不具合の混入は，開発側が業務側の作成した画面遷移図を正しく理解できていないまま UML 活動図を作成したことによると推測される．一方，場合 3 と場合 4 の不具合は，画面遷移図にエラー画面と戻り先画面への遷移に関する記入漏れが多数あったものである．これらの記入漏れは人的ミスによると考えられる.

今回の実験ではモデル作成の段階でも場合 3 と場合 4 に属する幾つかの不具合を発見できた．従って，記入漏れのような設計仕様の不具合は検証前の形式化の段階でも発見しやすいと考えられる．この点は形式化の利点の 1 つである．ただし，この種類の不具合を発見するには形式化を行う人が仕様をある程度理解している必要がある．

実験では，(e1)，(e2) に関する検査項目の他にもデッドロックの性質を CTL で記述して同様にモデル検査したところ，M1 の仕様にデッドロックが含まれていることを発見できた．そのデッドロックは，エラー画面を表示した後に他のどの画面へも遷移できないというものであった．

以上のように，提案法を用いて実アプリケーションの画面遷移図とフローチャート（UML 活動図）に対して画面遷移に関する幾つもの不整合を短時間で検出できた．このように提案法は，実際の Web アプリケーションを対象とした画面設計仕様を形式的に検証し信頼性を保証する上で効果的な手法であることが確認できた．

5　まとめ

本研究では，Web アプリケーション設計における画面遷移仕様とフローチャート間の整合性を形式的に検証する手法を提案した．具体的には，まずフローチャートが画面遷移仕様を満たすことを数理的に定義し，次にその定義と画面付きフローチャートの Kripke モデル上で CTL 式 (e1)，(e2) が成り立つことが等価であることを利用して，モデル検査を用いた整合性の検査手法を導いた．さらに，実際の Web アプリケーションの設計仕様に対して適用実験を行った結果，デッドロックを含む幾つかの不具合を短時間で検出でき，提案法の有効性が確認できた．

提案法では，画面遷移仕様とフローチャートの整合性を，画面遷移仕様に画面 a から画面 b への遷移があればフローチャートにその画面遷移に対応する遷移があること，かつ，フローチャートに画面 c から画面 d への画面遷移に対応する遷移があれば画面遷移仕様にその画面遷移があることとして定義した．従って，提案した整合性検査によって，画面遷移仕様の画面遷移に対応するフローチャートの遷移がない不具合や画面遷移仕様にない画面遷移に対応するフローチャートの遷移がある不具合を発見できる．

本研究の画面遷移に関する整合性の定義では，画面遷移仕様のある画面 a に対応した複数のフローチャートの状態があるとき，それらの状態間の関係については考慮していない．たとえば，同じ画面 a に対応するフローチャートの状態が S' と S'' という 2 つの集合に分割され，S' と S'' の間には遷移がないと想定すると，画面 a から画面 b への遷移に対応するフローチャートの遷移が一方の S' にあっても他方の S'' の状態からは画面 b に対応する状態へ到達できないという場合がある．現在の整合性の定義ではこのような場合を検出することはできないが，これを不具合とするべきか否かは意見の分かれるところである．ただし，Web アプリケーションの正当性を考える上でより複雑な画面遷移の仕様を検査できるようにすることは好ましい．たとえば上記の場合は，デッドロック検査で検出できる．他にも，「画面 a の次は画面 b でその次は画面 c」のような複数の画面を渡る画面遷移に対応するフローチャートの遷移系列があるか否かということも CTL 式で表現し検査できる．

今後の課題として，フレームやリンクなどの複雑な画面仕様を考慮したモデル化やより複雑な画面遷移の条件，データの依存関係，イベント処理などにも注目した Web アプリケーションの検証法を考えていきたい．また，今回の実験では与えられた Web アプリケーションの設計仕様から人手によってモデルおよび検査項目を作成したが，人手による形式化では誤りが生じる可能性がある．今後の課題として，形式化からモデル検査までの自動化にも取り組みたい．

参考文献

[1] Clarke, E. M., Grumberg, O., and Peled, D.: *Model Checking*, MIT Press, 1999.

[2] Holzmann, G. J.: The Model checker SPIN, *IEEE Trans. Softw. Eng.*, Vol. 23,No. 5(1997), pp. 279–295.

[3] Huth, M. R. A. and Ryan, M. D.: *Logic in Computer Science - Modelling and reasoning about systems*, Cambridge University Press, 2000.

[4] Kato, K., Yuen, S., Kato, D., and Agusa, K.: Web automaton: A behavioral model of Web applications based on the MVC Model, 日本ソフトウェア科学会ディペンダブルソフトウェア研究会 *(DSW'04)*, 2004, pp. 111–120.

[5] MacMillan, K. L.: *Symbolic Model Checking*, Kluwer Academic, 1993.

[6] NuSMV: http://nusmv.irst.itc.it/.

[7] Ricca, F. and Tonella, P.: Analysis and Testing of Web Applications, *Proc. of the 23rd International Conference on Software Engineering (ICSE 2001)*, 2001, pp. 25–34.

[8] Sciascio, E. D., Donini, F. M., Mongiello, M., and Piscitelli, G.: Web Applications Design and Maintenance Using Symbolic Model Checking, *Proc. of the 7th European Conference on Software Maintenance and Reengineering (CSMR 2003)*, 2003, pp. 63–72.

[9] UML Revision Taskforce: OMG UML Specification v. 1.4., *Object Management Group*, (2001).

[10] UPPAAL: http://www.uppaal.com/.

[11] WWV'05: The 1st Int'l Workshop on Automated Specification and Verification of Web Sites: http://www.dsic.upv.es/workshops/wwv05/.

画面遷移仕様のモデル検査
(算譜科学研究速報)
発行日：2005 年 1 月 20 日
編集・発行：独立行政法人産業技術総合研究所関西センター尼崎事業所
システム検証研究センター
同連絡先：〒661-0974 兵庫県尼崎市若王寺 3-11-46
e-mail：informatics-inquiry@m.aist.go.jp

Model Checking for Verifying Specification of Page Flow (in English)
(Programming Science Technical Report)
January 20, 2005
Research Center for Verification and Semantics (CVS)
AIST Kansai, Amagasaki Site
National Institute of Advanced Industrial Science and Technology (AIST)
3-11-46 Nakouji, Amagasaki, Hyogo, 661-0974, Japan
e-mail : informatics- inquiry@m.aist.go.jp