画面遷移を用いたWebアプリケーションのモデル化とＳＰＩＮによる検証

本間　圭[†1]　　高橋　薫[†2]　　富樫　敦[†3]

宮城大学大学院事業構想学研究科　仙台電波工業高等専門学校　宮城大学大学院事業構想学研究科

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処理は増加の一途をたどっている．

しかし，アプリケーションの設計に焦点を当てると，構築プロセスの自動化や省力化と比較して，設計段階の自動化はあまり行われておらず，設計に対する整合性の確認は人手によって行われているのが現状である．

そこで，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焦点をあて，画面遷移とシステム環境をそれぞれ（非決定性）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を提案する．

また，サンプルのアプリケーションに対して画面遷移およびシステムの変数を割り当てたモデルを作成し，モデル検証ツールのＳＰＩＮ [1] を用いることにより設計モデルに不具合がないことを確認した．

2　画面遷移図とシステム状態のモデル

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例え，画面遷移を状態遷移と見立てることにより有限オートマトン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，受理状態というものは通常考えないとすると，画面遷移のオートマトンを以下のように表現できる．

$$M_G = \langle Q_G, \Sigma, \delta_G, q_{0G} \rangle \tag{1}$$

ここで，Q_G: 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_G: 動作関係（画面遷移）

$(\delta_G \subseteq Q_G \times \Sigma \times Q_G)$

q_{0G}: 初期状態（トップ画面）$(\in Q_G)$

ここで，トランザクションを扱うWebアプリケーションを考えると，画面と同様にシステム内部の状態が重要である．画面により入力された情報を内部に保持し，その変数の集合によってシステム内部の状態が決定する．画面の遷移は多くの場合，押下されたボタンのみで判断できるものでなくシステム内部の状態も考慮してその挙動が決まる．このように考えた場合，システム環境も有限オートマトンとして以下のように表現する．

$$M_e = \langle Q_e, \Sigma, \delta_e, q_{0e} \rangle \tag{2}$$

ここで，Q_e: 状態の有限集合

（変数の集合が表す状態）

Σ: 入力記号の有限集合

（システムに対するアクション）

δ_e: 動作関係（変数の変更）

$(\delta_e \subseteq Q_e \times \Sigma \times Q_e)$

q_{0e}: 初期状態（$\in Q_e$）

上記 (1)，(2) の有限オートマトンの直積をとることによってシステム全体のオートマトンを以下のように表現する．

$$M = M_G \times M_e = \langle Q, \Sigma, \delta, q_0 \rangle \tag{3}$$

ここで，$Q \subseteq Q_G \times Q_e$

$\delta((q_G, q_e), a, (q'_G, q'_e)) \Leftrightarrow$

$\delta_G(q_G, a, q'_G)$ かつ $\delta_e(q_e, a, q'_e)$

$q_0 = (q_{0G}, q_{0e})$

ただし，　$q_G, q'_G \in Q_G$, $q_e, q'_e \in Q_e$, $a \in \Sigma$

3　SPINを用いたWebアプリケーションの検証

前述したモデルを検証ツールのSPINを利用して検証する．SPINはモデル検証ツールとしていくつかの機能がある．ここでは以下の内容を検証することとする．

1. 画面遷移とシステム環境の遷移を連動させたとき，デッドロックが生じないことを確認する．
2. テストケースを用いて，画面遷移およびシステム環境が正しく稼働することを確認する．

Model Checking On Page Transitions and System States Using SPIN

†1Kei HOMMA
Graduate School of Project Design, Miyagi University
†2Kaoru TAKAHASHI
Sendai National College of Technology
†3Atsushi TOGASHI
Graduate School of Project Design, Miyagi University

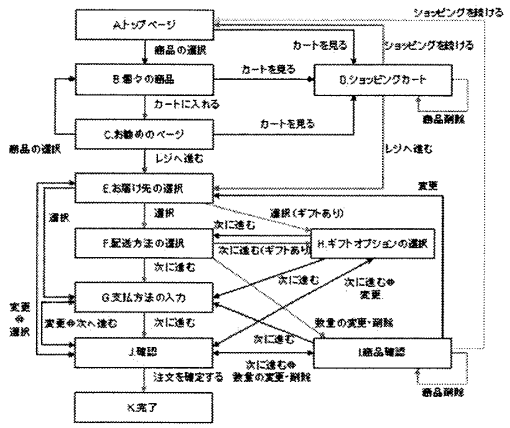
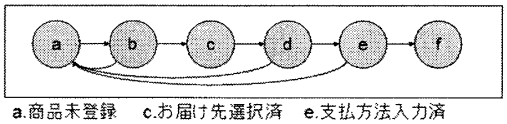


図 1: 画面遷移図



a.商品未登録　c.お届け先選択済　e.支払方法入力済
b.商品登録済　d.配送方法選択済　f.注文確定済

図 2: システム環境のオートマトン

1．は SPIN の到達性の解析を利用することによって表現可能となる．つまり，デッドロックが発生しないことにより確認する．2．は時相論理を用いた SPIN の LTL モデル検証機能を利用し，エラーとならないこと，もしくは期待通りに反例が出力されることにより確認する．

Amazon.com のオンラインショップを例に，モデル化および検証を行う．画面遷移を図 1 に示す．この画面遷移では，ユーザは商品を選択し，レジへ進んだ後にお届け先や支払方法等の必要な項目を入力して完了画面へと遷移する．この画面遷移には特徴が 2 つある．一つ目は，システム環境によって画面から遷移可能な次画面が制御されている．例えば，商品が選択されていない状態で「ショッピングカート」画面から「お届け先の選択」画面へ遷移はできない．二つ目は，ある画面のボタンを押下した際の次画面は一意には決まらない．例えば，お届け先の選択画面の「次へ」ボタンは「配送方法の選択」画面,「ギフトオプションの選択」画面,「確認」画面へ遷移する可能性がある．

この複数遷移を制御するのがシステム環境である．システム環境のオートマトンを図 2 に示す．システム環境の遷移は画面の遷移と連動している．例えば，「個々の商品」画面の「カートへ入れる」ボタンを押下することにより，画面は「お勧めのページ」画面へ遷移する際に,「商品未登録」状態が「商品登録済」状態へ遷移すると考える．

検証の方法としては，まず，上記の 2 つのオートマトンをプロセスとして SPIN の Promera で記述する．記述方法の詳細は割愛するが，2 節で述べた通り，画面遷移のオートマトンとシステム環境のオートマトンの直積をシステム全体のオートマトンとして表現できるよう，2 つのプロセスをチャネルを用いて連動させた形で表現する．

設計全体の整合性の確認は上記の通り，SPIN の到達性の解析で行う．SPIN で検証器を生成し，デッドロックが発生しないことを確認した．次に，テストケースを用いる場合は複数のケースが考えられるが，ここでは 2 つ例を挙げる．1 つは「確認」画面から「トップページ」画面へ遷移が可能であることを否定の条件式を用い，その反例が出力されることにより確認する（式 4）．もう一つは”「個々の商品」画面かつ「支払方法入力済」状態であるということがないという事実”を確認する（式 5）．いずれも SPIN で検証器を生成しモデル検査を実行し，期待通りの結果を得ることができた．

$$\neg[](p \rightarrow <> q) \tag{4}$$

ここで，　p ==「確認」画面
　　q ==「トップページ」画面

$$[]\neg\ (p\ \&\&\ q) \tag{5}$$

ここで，　p ==「個々の商品」画面
　　q ==「支払方法入力済」状態

4　まとめ

以上により，画面遷移およびシステム環境を用いてシステム全体の状態を表現し，SPIN によって一定のモデル検証が可能であるということを示した．

大規模な Web アプリケーションの構築では，複数の担当者が画面設計を行う．画面遷移のみを確認した場合には整合性が確保されているように見えるが，実装した段階で画面間の不整合が発生しやすい．当検証方法はシステム構築の早い段階で行うことが可能なため，実装やテスト行程で問題が発見される場合と比較して工数の大幅な削減が期待できる．

参考文献

[1] 中島震，“SPIN モデル検査”，近代科学社，2008.

[2] 崔銀惠，河本貴則，渡邊宏，“画面遷移仕様のモデル検査”，産業技術総合研究所，2005.