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

2008－EMB－10 (13)
2008／11／28

状態遷移表に基づくモデル検査の支援環境

小池 隆

富士ソフト株式会社

モデル検査はソフトウェアの設計品質向上に有効な技術だが，モデル記述言語と時相論理を習得して検査モデルと制約条件を記述しなければならないことが，産業界への普及の妨げとなっている．
そこで，状態遷移表と付加的な表から制約条件まで含む検査モデルを自動生成するモデル検査支援環境を作成し，複数の組込みソフトウェア開発プロジェクトへの適用によって評価した．

Model Checking Support Environment based on State Transition Matrix

Takashi Koike

FUJISOFT INCORPORATED

Model Checking is effective to improve quality of software design. But, it requires knowledge of model description language and temporal logic to describe models and properties by hand, which make it difficult to be widely practiced in the industry.
We developed Model Checking Support Environment to automatically generate models including properties to be checked, from State Transition Matrix and some additional tables. We also applied it to some embedded software development projects and evaluated.

1. はじめに

コンピュータが普及し，社会基盤としてのソフトウェアの比重が高まる一方において，ソフトウェアの不具合が大きな社会問題へと発展してしまうことがある．
ソフトウェアの品質・信頼性に対する要求が高まる中で注目されているのが形式手法である．特に，振舞いを網羅的に検証するモデル検査は，組込みソフトウェア開発への適用が期待されている．
しかし，実際にモデル検査を実施するには，検査モデルと制約条件を手作業で記述しなければならないため，モデル記述言語と時相論理の習得が必要になる．このことが開発現場へのモデル検査技術の普及の妨げとなっている．
そこで，モデル記述言語と時相論理の習得が不要なモデル検査支援環境を構築し，複数の組込みソフトウェア開発プロジェクトに適用して評価した．

2. モデル検査支援環境

モデル検査支援環境は，開発現場のエンジニアが手作業でモデル記述言語や時相論理式を用いた記述をすることなく，モデル検査を実施できる環境を提供する．
そのために，通常の開発作業において作成される設計ドキュメントから，モデル検査に必要なプログラムや論理式を自動生成する．以下にその詳細を記す．

2.1 状態遷移表

モデル検査の対象となる状態遷移系のモデル化手法には，UML のステートマシン図や状態遷移表がある．どちらも開発現場では多く使用されているが，モデル検査支援環境への入力ドキュメントの記法として状態遷移表を採用した．その理由は，表形式のために設計の「モレ」や「ヌケ」が少なく，モデル検査実施以前の段階で品質がある程度確保されていることが期待できるためである．
状態遷移表には以下の 2 つの形式がある．

- 現在の状態とイベントのマトリックスを作成し，各交点のセルの中に，実行するアクションと遷移先の状態を記述する．
- 現在の状態と次の状態のマトリックスを作成し，各交点のセルの中に，その遷移を発生させるイベントと実行するアクションを記述する．

組込みソフトウェア開発の現場では主に前者の形式が使用されるため，本研究においても前者を採用する．
表 1 に状態遷移表の例を示す．
この例では，列方向に状態を，行方向にイベントを列挙しているが，行と列は逆でも良い．
状態とイベントの交点（セル）には，遷移先の状態名と実行するアクション名を記入する．この例では上段に遷移先を，下段にアクションを記入しているが，遷移先とアクションの上下は逆でも良い．
また，「／」は無視セル，「×」は不可セルを表す．無視セルは，当該状態で当該イベントが発生しても，アクションと遷移を実行しないことを表

す．不可セルは，当該状態では当該イベントが発生し得ないことを表す[1]．

表1　状態遷移表

	CDなし	再生中	停止中
CD挿入	再生中 CD受入 再生開始	×	×
再生ボタン	／	／	再生中 再生開始
停止ボタン	／	停止中 再生終了	／
イジェクト	／	CDなし CD排出	CDなし CD排出

表1の状態遷移表では，「CDなし」状態で「CD挿入」イベントが発生したときには，「CD受入」アクションと「再生開始」アクションを実行し，「再生中」状態に遷移する．また，「CDなし」状態で「再生ボタン」イベントが発生しても何もせず，「再生中」状態で「CD挿入」イベントは発生し得ない．

2.2 プロパティ表

状態遷移表だけでは，システムが満たすべき性質までは記述されない．

モデル検査では，通常，システムが満たすべき性質を時相論理式で記述する．しかし，ソフトウェア開発の現場において時相論理の知識は一般的ではない．また，1つのシステムが満たすべき性質は多数あるため，難解な時相論理式を大量に記述しなければならないことになり，現実的ではない．

そこで本研究では，システムが満たすべき性質を簡易に列挙するための「プロパティ表」を考案した．

プロパティ表は，システムの構成要素と，システムの状態とのマトリックスに，その構成要素の取るべき値を制約条件として記述するものである．

表2にプロパティ表の例を示す．列方向には，表1の状態遷移表に対応した状態が並んでいる．行方向には，システムの構成要素として「CD」と「モータ」が挙げられている．例えば「再生中」状態では，「CD」は「あり」で，「モータ」は「回転」でなければならないことを表している．

表2　プロパティ表

	CDなし	再生中	停止中
CD	なし	あり	あり
モータ	停止	回転	停止

モデル検査によって，プロパティ表に記述された制約条件を満たすことが検査される．もしも「再生中」状態で「CD」が「なし」になっていたら，不具合として検出される．

2.3 アクション表

表1の状態遷移表のアクション欄に記述されるのはアクションの名前であり，それだけでは，実行するアクションの具体的な内容は不明である．そのため，表2のプロパティ表に挙げられたシステムの構成要素の値が，どのアクションでどう変化するのかを，別途明確にする必要がある．そこで本研究では「アクション表」を考案した．

アクション表では，状態遷移表に記載されたアクションの各々について，その実行に伴ってシステムの構成要素の値がどう変化するかを記述する．さらに，アクション表には，そのアクションの実行前に成立していなければならない条件と，実行後に成立していなければならない条件を記述することができる．それらは，プロパティ表に記述された制約条件と同様に，モデル検査において検査される．

表3にアクション表の例を示す．「＝」は等号を，「←」は代入記号を表す．

例えば「再生開始」アクションの実行前には，「CD」が「あり」で「モータ」は「停止」でなければならない．「再生開始」アクションを実行すると「モータ」は「回転」に変化する．

表3　アクション表

	事前制約	処理内容	事後制約
再生開始	CD＝あり モータ＝停止	モータ←回転	－
再生終了	CD＝あり モータ＝回転	モータ←停止	－
CD受入	CD＝なし	CD←あり	－
CD排出	CD＝あり	CD←なし	－

もしも，モデル検査において「CD」が「なし」のときに「再生開始」アクションを実行する動作例が見つかると，事前制約違反の不具合として報告される．

2.4 条件判定表

システム開発の現場で作成される状態遷移表では，アクションや遷移の実行に条件が付与される場合が多い．それによって，セル内の記述量が増える反面，状態遷移表上の状態数を減らすことができる．

そこで，モデル検査支援環境では，状態遷移表のアクション欄と遷移先欄において，角括弧［］内に条件を記述することを可能とした．

表4の状態遷移表では，「電源OFF」状態で「電源ON」イベントが発生したときに，「CDあり」の条件が成り立つ場合には「再生開始」アクションを実行して「再生中」状態に遷移する．「CDなし」条件が成り立つ場合には，何もせずに「待機

中」状態に遷移する．

表 4　条件判定付きの状態遷移表

	電源ＯＦＦ	待機中	
電源ＯＮ	[ＣＤあり]再生中 [ＣＤなし]待機中	×	
	[ＣＤあり]再生開始		

状態遷移表で［］内に記述されるのは，条件判定の名前だけであり，その条件の真偽がどのようにして決まるのかという判定内容までは記述されない．

そこで，条件判定の内容は，「条件判定表」を用いて定義する．判定内容欄には，システムの構成要素を変数として用いた条件式の形式で記述する．

表 5 の条件判定表では，システムの構成要素「ＣＤ」の値が「あり」の場合に，条件判定「ＣＤあり」が成立し，「ＣＤ」の値が「なし」の場合に，条件判定「ＣＤなし」が成立する．

表 5　条件判定表

	判定内容
ＣＤあり	ＣＤ＝あり
ＣＤなし	ＣＤ＝なし

2.5 検査モデルの自動生成

モデル検査の実施にあたっては，モデル検査器が必要となる．本研究では，ベル研究所で開発され，オープンソースソフトとして公開されている SPIN を使用した[2]．

SPIN は，独自のモデル記述言語 PROMELA（PROcess MEta LAnguage）で記述したモデルを検査する．そのため，本研究においては，前述の状態遷移表，プロパティ表，アクション表，条件判定表から PROMELA 言語で記述したモデル検査プログラムを自動生成するツールを開発した．

SPIN でモデル検査をする場合，システムが満たすべき性質を時相論理の一種である LTL（線形時相論理）を用いて記述するのが普通である．しかし，開発現場のエンジニアには LTL は馴染みが浅い．そこでモデル検査支援環境は，プロパティ表に記述された制約条件と，アクション表に記述された事前制約／事後制約を，PROMELA プログラムの中に assert 文として出力する．それによって，別途 LTL を記述することなく，自動生成した PROMELA ファイルのみで SPIN によるモデル検査を実施することができる．

モデル検査において，システムが満たすべき性質を満たしていないことが検出された場合，その制約違反に至る動作シーケンスが「反例」として報告される．

表 1 の状態遷移表，表 2 のプロパティ表，表 3 のアクション表から生成した PROMELA プログラムを使用してモデル検査を実施すると，プロパティ表に記述した制約への違反により反例が出力される．

このときの反例シーケンスを解析すると表 6 のようになる．

表 6　反例シーケンス

状態	イベント	アクション	プロパティ	
			ＣＤ	モータ
ＣＤなし			なし	停止
	ＣＤ挿入		↑	↑
		ＣＤ受入	あり	↑
		再生開始	↑	回転
再生中			↑	↑
	イジェクト		↑	↑
		ＣＤ排出	なし	↑
ＣＤなし			↑	↑

「ＣＤなし」状態で「ＣＤ挿入」イベントが発生し，「再生中」状態に遷移した後，「イジェクト」イベントが発生し「ＣＤなし」状態に遷移する．「ＣＤなし」状態では「モータ」の値が「停止」でなければならないという表 2 のプロパティ表の記述に反し，反例シーケンスでは「回転」となっている．

この不具合は，「再生中」状態に「イジェクト」イベントが発生したときのアクションに「再生終了」を追加することにより解消する．

2.6 LTL 生成支援

前述の通り，システムが満たすべき性質をプロパティ表とアクション表に記述することにより，モデル検査プログラムに assert 文として出力される．

しかし，検査したい性質の中には，特定の状態やアクションとの関係で定義できないものもある．そのような場合には LTL を使用する必要があるため，LTL の自動生成を支援する環境を用意した．

自動生成できるのは，以下の 2 つの性質に関する LTL である．

●状態到達性
●プロパティ値到達性

状態到達性とは，ある状態から，別のある状態へ遷移することができるかどうかということである．簡単な状態遷移表であれば，目視により確認することができる．しかし，状態遷移に条件が付けられている場合には，その条件が成立し得るかどうかによって，遷移し得ない状態がでてきてしまうこともあるため，モデル検査で動的かつ網羅的に検査することは意味がある．

プロパティ値到達性は，システムの構成要素が，ある任意の値を取り得るかどうか，ということで

ある．例えば，自動販売機が商品を1つも残さずに売り切ることができるかどうかを検査するために，システムの構成要素である「商品在庫数」が0になり得るかどうかを検査する．

2.7 反例解析支援

SPINは反例をtrailファイルとして出力する．しかし，trailファイルに出力されるのはPROMELAプログラム上の実行シーケンスや変数である．表6のように状態遷移表上の「状態」や「イベント」「アクション」が出力されるのではない．そのため，trailファイルを読んで状態遷移表上の不具合を発見するには，自動生成されたPROMELAプログラムと状態遷移表との対応関係を理解しなければならなく，それでは状態遷移表からPROMELAプログラムを自動生成することの意味が半減してしまう．

そこで，反例として得られたtrailファイルを入力し，状態遷移表と対応付けてシーケンスをグラフィカルに表示する反例解析支援環境を用意した．

これによって，制約違反に至る反例を，状態遷移表上のイベント発生とアクション実行，状態遷移のシーケンスとして確認することができ，状態遷移表のデバッグが可能になる．

図1に，反例解析支援環境を使用した反例シーケンスの表示例を示す．ベース画面に検査対象の状態遷移表が表示され，ポップアップ画面に反例のシーケンスが表示される．シーケンス上のカーソル位置を移動すると，対応する状態遷移表のセルが強調表示される．

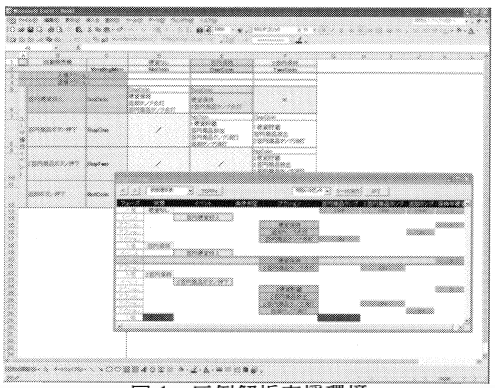


図1　反例解析支援環境

2.8 モデル検査の流れ

モデル検査支援環境を使用したモデル検査の流れは図2のようになる．

モデル検査で不具合が発見された場合には，反例解析によって原因を明らかにし，表を修正した後，モデル検査プログラムを自動生成して直ちに再検査することができる．

設計と検証のサイクルの高速化は，モデル検査支援環境の大きなメリットである．

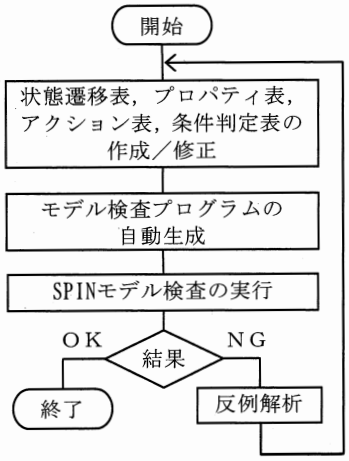


図2　モデル検査の流れ

3. フォールスアラームへの対処

モデル検査支援環境を実際の開発プロジェクトで使用する中で，しばしばフォールスアラームが発生した．フォールスアラームとは，実際には設計上の問題はないにもかかわらず，モデル検査において不具合が報告されることである．

フォールスアラームが発生したときには，自動生成したPROMELAプログラムを，反例が出力されないように手作業で修正した．しかし，これは効率の悪い作業である．

3.1 フォールスアラームの発生原因

そこで，フォールスアラームの発生事例を分析し，主に以下の2つの原因によってフォールスアラームが発生することが分かった．

- 自動生成されるモデル検査プログラムでは，イベントはすべて非決定的に発生させていた．しかし，あるイベントが特定の条件下でのみ発生することを前提に設計されている場合があり，そのような場合において，想定外のイベント発生によるアクション実行または状態遷移が，プロパティ制約違反を引き起こしていた．
- システムの構成要素の値の変化は，アクションの中で能動的にのみ行うものと考えていた．しかし実際には，システムの外部から受動的に変化することがあり，そのような変化をモデル化できないために，プロパティ制約違反を引き起こしていた．

これらは，検査対象のシステム自体に内在する問題ではなく，システムの外部環境によって引き起こされた問題である．一般に，モデル検査を適用するには検査対象の外部環境も含めて「閉じた系」にしなければならない[3]．しかし，状態遷移

表とアクション表のみでは外部環境までモデル化できないために「閉じた系」とならず，フォールスアラームが発生したと考えられる．

3.2 イベント表

状態遷移表で記述されたシステムの外部環境として最低限，以下の2つをモデル化することによって，「閉じた系」を構成することが可能であると考えた．

- イベントの発生条件
- イベントの発生に伴って生じる，システムの構成要素の値の変化

これらを記述するために「イベント表」を考案した．

表7にイベント表の例を示す．行方向にイベントを列挙し，それぞれについて「発生条件」と「作用内容」を記述している．

表7　イベント表

	発生条件	作用内容
電源ボタン押下	電源ボタン＝上	電源ボタン←下
電源ボタン押上	電源ボタン＝下	電源ボタン←上
タイマ割込み	割込み＝許可	
ディスク挿入	ディスク＝なし	ディスク←あり

例えば，「電源ボタン押下」イベントは電源ボタンが上がっているときにしか発生せず，「電源ボタン押上」イベントは電源ボタンが下がっているときにしか発生しない．これによってトグルボタン操作によるイベント発生を忠実にモデル化している．また，「タイマ割込み」イベントは割込みが許可されていなければ発生しない．

作用内容には，イベントの発生に伴う，システムの構成要素の変化を記述する．「ディスク挿入」イベントが発生すると，システムの構成要素「ディスク」は「あり」になる．

イベント表の記述内容を含んだモデル検査プログラムを自動生成できるようにした結果，フォールスアラームの発生をなくすことができた．

4. プロジェクトへの適用

4.1 適用対象

作成したモデル検査支援環境を，表8に示す組込みソフトウェア開発プロジェクトに適用して評価した．

モデル検査の導入フェーズはプロジェクトによって異なった．プロジェクトの開始当初からモデル検査の実施を意識した事例もあれば，既にコーディング以降のフェーズに入っているプロジェクトに対して，後追いでモデル検査を実施した場合もあった．

表8　適用対象プロジェクト

項番	プロジェクト内容
1	携帯電話の内蔵アプリケーション
2	産業機器の画面表示モジュール
3	デジタルＴＶの通信モジュール
4	車載機器の内蔵アプリケーション
5	産業機器の通信モジュール
6	車載機器の通信制御モジュール
7	光学機器の制御モジュール
8	通信機器の通信モジュール

いずれの事例においても，プロジェクトメンバによるレビューが完了している状態遷移表を検査対象とした．

状態遷移表の抽象度はプロジェクトによって任意とした．そのため，基本設計レベルの状態遷移表から，プログラム設計レベルの状態遷移表までが対象となった．

検査した状態遷移表の数は，プロジェクトによって1つから4つである．

状態遷移表の規模は，状態数の最大が80，イベント数の最大が159である．多くは，状態数が10程度，イベント数が30以下である．

モデル検査の実施は，筆者らが行う場合と，開発現場のエンジニアが自分自身で行い，筆者らが支援する場合があった．

4.2 適用結果

表8に示したすべてのプロジェクトにおいて，各々2件から3件程度の不具合が発見された．

モデル検査で発見された不具合の，内容による集計を表9に示す．なお，この件数にはフォールスアラームは含んでおらず，すべての不具合が設計の修正を必要としたものである．

表9　検出された不具合

不具合内容	件数
プロパティ制約違反	１７
アクション事前制約違反	2
状態未到達	1
プロパティ値未到達	1
計	２１

最も多く発見されたプロパティ制約違反は，システムの構成要素が，プロパティ表に記述された制約値から外れた値をとるという不具合である．遷移先状態の単純な記載ミス，アクションの実行条件の記載漏れ，実行するアクションの間違い，条件判定の内容の間違い等の原因によって生じた．

アクション事前制約違反は，アクション表の事前制約欄に記述された条件を満たさないという不

具合である．実行するアクションを変更し，遷移先の状態を新たに追加するという，比較的大きな修正を必要とする誤りによって生じた．

状態未到達は，本来到達可能でなければならない状態に到達できないという不具合である．遷移先状態の単純な記載ミスによって生じた．

プロパティ値未到達は，システムの構成要素が取り得るべき値を取り得ないという不具合である．条件判定の内容（計算アルゴリズム）の間違いによって生じた．

状態未到達とプロパティ値未到達の不具合は，L自動生成したLTLを用いた検査で発見された．

5. 考察

5.1 不具合の検出能力

表9に示した通り，多くの不具合がプロパティ制約違反として発見された．単純な記載ミスから，比較的根の深いものまで，発生原因は様々でありながら，多くの不具合を発見することができた．システムが満たすべき性質を，システムの状態とその状態におけるシステムの構成要素のとるべき値によって規定するという提案手法の有効性を確認することができた．

5.2 課題

プロジェクトへの導入において，いくつかの課題が明らかになった．

5.2.1 状態遷移表の抽象度・粒度

既述のとおり，状態遷移表の抽象度・粒度はプロジェクト毎に任意とした．1つの状態遷移表の中で抽象度・粒度が一定し，整合性が取れている限りにおいては，特に問題は生じなかった．しかし，1つの状態遷移表の中で抽象度・粒度が一定しない事例があり，その場合は不整合による不具合が検出された．

複数のメンバで1つの状態遷移表を作成する場合には，状態遷移表の抽象度・粒度を統一するために，状態遷移表の記述ルールをあらかじめ策定することが必要である．

5.2.2 プロパティ表の記述

提案手法では，システムが満たすべき性質を，システムの状態とその状態におけるシステムの構成要素のとるべき値によって規定する．しかし，そのような形式で制約条件を記述することに馴染みの浅いエンジニアは，プロパティ表の作成に困難を伴う場合があった．

特に画面表示系の事例において，システムの構成要素として何を挙げれば良いか分からない場合が多かった．

一方，機械制御系の事例においては，ハードウェアコンポーネントをシステムの構成要素とすることにより，その制約条件の記述を比較的容易に行うことができた．

今後，事例を増やしていくなかで，プロパティ表の記述ノウハウを蓄積することによってこの課題を解決したい．

5.2.3 イベント表の記述

検査モデルを「閉じた系」とするためにイベント表を作成するということは，通常の設計作業では行われないため，その必要性を理解させることが困難だった．

また，組込みソフトウェアでは，本来発生し得ないはずのイベントが，例えばセンサーの異常によって発生するという事態まで考慮しなければならないことがある．イベントの発生条件を，設計者の「期待」によってあまりにも厳格に規定してしまうと，想定外のイベントに対して正しく対処しないという不具合をモデル検査では発見できなくなってしまう．

どこまでイレギュラーなイベントが発生してもシステムが正しく対処しなければならないのか，状態遷移設計とモデル検査の実施以前に，要件定義の段階で明確にしておかなければならない．

6. おわりに

状態遷移表と付加的な表から，制約条件まで含むモデル検査モデルを自動生成するモデル検査支援環境を作成した．複数の組込みソフトウェア開発プロジェクトへ適用し，設計上の不具合発見に提案手法が有効であることを確認した．

今後は，プロジェクトへのモデル検査支援環境の適用事例を増やすことによって，適用ノウハウを蓄積し，手法をさらに洗練したい．

参考文献

[1] 渡辺政彦，拡張階層化状態遷移表設計手法 Ver.2.0—Embedded SE のための設計手法，キャッツ，1998

[2] G.J. Holzmann, The Spin Model Checker: Primer and Reference Manual, Addison-Wesley, 2003

[3] 中島震，SPIN モデル検査—検証モデリング技法，近代科学社，2008