2009－SE－163 (39)
2009／3／19

状態遷移表のモデル検査における LTL 検証パターン

矢野 恭平　　小池 隆

富士ソフト株式会社

ソフトウェア設計の品質向上の技術として形式手法，特にモデル検査が注目されているが，産業界への普及のためには，開発者にとって使いやすいツールの提供が必要である．そこで，筆者らは状態遷移表に基づくモデル検査の支援環境を構築した．

本稿では，状態遷移表のモデル検査で使用される LTL（Linear Temporal logic）の検証パターンについて議論し，モデル検査支援環境による，パターン化された LTL 検査式の自動生成と自動検査の有用性について評価する．

LTL Verification Patterns for Model Checking of State Transition Matrix

Kyohei Yano and Takashi Koike

FUJISOFT INCORPORATED

Formal Methods, especially Model Checking, have lately drawn attention as effective methods to improve quality of software design. Although, to be widely practiced in the industry, easy to use tools for developer is necessary. So, we developed Model Checking Support Environment based on STM(State Transition Matrix).

In this paper, we discuss LTL(Linear Temporal logic) verification patterns to be used to model check the STM, and evaluate the pattern-based LTL formula auto-generation and batch test functionality of Model Checking Support Environment.

1. はじめに

ソフトウェアの品質・信頼性に対する要求が高まる中で，振舞いを網羅的に検証するモデル検査技法は，特に組込みソフトウェア開発において適用が進められている．

モデル検査の実施には，SPIN[1]等のモデル検査器が使用される．しかし，モデル検査を実施するには，PROMELA（PROcess MEta LAnguage）等の特殊なモデル記述言語を用いて検査モデルを記述し，LTL（Linear Temporal logic）等の難解な時相論理を用いて制約条件を記述しなければならない．このことが産業界へのモデル検査技術の普及の妨げとなっている．

そこで筆者らは，組込み系のソフトウェア開発現場で作成されることの多い状態遷移表[2]と付加的な表から制約条件まで含む検査プログラムを自動生成するモデル検査支援環境[3]を作成し，開発プロジェクトへ適用した．

モデル検査支援環境では，システムに求められる性質を，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取るべき値によって定義する．また，アクションの実行前／実行後の，システムの構成要素の取るべき値を制約として定義することもできる．これらの制約条件は PROMELA プログラム中に assert 文として出力され，モデル検査器 SPIN によって，制約違反が生じる動作シーケンスがないか網羅的に検査される．

この仕組みによって，システムに求められる性質を LTL 式で記述することなしにモデル検査を実施することができる．

しかし，システムに求められる性質の中には，特定の状態やアクションとシステムの構成要素との関係による定義には向かないものもあり，そのような場合には LTL を使用する必要がある．

そこで本研究では，LTL による検証内容のパターン化を試み，パターンに基づく LTL 検査式を状態遷移表から自動生成する LTL 検査支援機能を作成し，その有用性を検証する．

2. 関連研究

M. B. Dwyer ら[4]は，仕様の記述を収集して分類することによって，図 1 に示す性質パターンを提唱した．

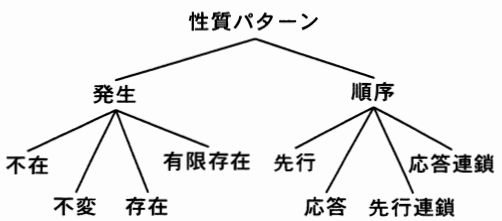

図 1　Dwyer の性質パターン

金井と岸[5]は，以下に示すUMLの特定の構造に対して，その構造において重要な性質を検証するためのパターン化を行った．

- 1-1メッセージ送受信構造
- 1-Nメッセージ送受信構造
- メッセージ送受信連鎖構造
- クラスの状態管理構造
- リソースの取り合い構造

西[6]は，状態遷移モデルのテストを，状態遷移パステストと状態遷移マトリクステストに分類した．状態遷移パステストは，イベントを順番に発生させることによって状態が正しく遷移することを確認する．状態遷移マトリクステストでは，イベントと状態の組合せを網羅的に確認する．

小池ら[7]は，習得困難なLTLをソフトウェア開発現場に導入するために，LTL検査式の図示記法を考案した．

3. モデル検査支援環境

モデル検査支援環境は，状態遷移表と付加的な表から，SPINモデル検査のためのPROMELAプログラムを生成する．以下にモデル検査支援環境の概要を記す．

3. 1 状態遷移表

状態遷移表は，現在の状態とイベントのマトリクスを作成し，各交点のセルの中に，上段には遷移先を，下段には実行するアクションを記述する．

表1に状態遷移表の例を示す．

表1　状態遷移表

		状態		
		CDなし	再生中	停止中
イベント	CD挿入	再生中 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	／
	排出ボタン	／	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，「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（無視セル），「再生中」状態で「CD挿入」イベントは発生し得ない（不可セル）．

無視セルは「／」，不可セルは「×」で示す．

3. 2 プロパティ表

プロパティ表は，システムに求められる性質を簡易に列挙するための表である．システムの構成要素について，その初期値と許容される値を定義し，さらに，システムの各状態において取るべき値を制約条件として記述する．

表2にプロパティ表の例を示す．

表2　プロパティ表

		初期値	許容値	状態
				停止中
構成要素	CD	なし	あり，なし	あり，なし
	モータ	停止	回転，停止	回転

表2のプロパティ表では，構成要素「CD」の初期値は「なし」で，許容される値は「あり」と「なし」である．「停止中」状態における制約条件として，「CD」は「あり」または「なし」で，「モータ」は「回転」でなければならない．

3. 3 アクション表

アクション表は，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，そのアクションの実行前／実行後に成立していなければならない制約条件も記述することができる．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．

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排出	CD＝あり	CD←なし	－

表3のアクション表では，例えば「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．その実行前には，「CD」が「あり」で「モータ」は「停止」でなければならない．

3. 4 条件判定表

状態遷移表では，アクションや遷移の実行にガード条件を付与することができる．表4にガード条件付きの状態遷移表を示す．

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　ガード条件付きの状態遷移表

		状態		
		電源ＯＦＦ	待機中	
イベント	電源ＯＮ	[ＣＤあり]再生中 [ＣＤなし]待機中	×	
		[ＣＤあり]再生開始		

ガード条件の判定内容は，条件判定表で定義する．判定内容欄には，プロパティ表で定義された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 に条件判定表の例を示す．

表 5　条件判定表

	判定内容
ＣＤあり	ＣＤ＝あり
ＣＤなし	ＣＤ＝なし

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．

3. 5 イベント表

イベント表には，イベントの発生条件と，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（作用内容）を記述する．表 6 にイベント表の例を示す．

表 6　イベント表

	発生条件	作用内容
ディスク挿入	ディスク＝なし	ディスク←あり
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	

表 6 のイベント表では，例えば，「ディスク挿入」イベントはシステムの構成要素「ディスク」が「なし」のときしか発生せず，「ディスク挿入」イベントが発生すると，「ディスク」は「あり」に変化する．

イベント表を作成し，状態遷移表の外部環境を適切にモデル化することによって，生成されるモデル検査プログラムを「閉じた系」とし，フォールスアラームの発生を抑止することができる．

さらに，イベントの発生条件を指定することによって，無視セルと不可セルを区別し，不可セルへの到達を不具合として検出することが可能になる．

3. 6 検査プログラムの自動生成

モデル検査支援環境は，状態遷移表，プロパティ表，アクション表，条件判定表，イベント表から，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．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／事後制約は，PROMELA プログラム中に assert 文として出力されるため，別途 LTL 検査式を記述しなくても，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．

4. LTL 検査支援機能

システムに求められる性質の中には，プロパティ表とアクション表だけでは定義できないものもあり，そのような場合に LTL を用いた検査を実施する必要がある．

しかし，ソフトウェア開発現場において LTL は日常的に使用されてはいないため，モデル検査を導入するにあたって大きな障壁となってしまう．そこで本研究では，LTL を用いた検査を支援するために，モデル検査支援環境上に LTL 検査支援機能を作成した．

4. 1 LTL 検査式の自動生成

LTL 検査支援機能は，モデル検査支援環境で作成した状態遷移表と付加的な表から，特定の検証パターンに基づく LTL 検査式を LTL 検査項目表として自動生成する．これにより，LTL の知識を必要とせずに，LTL 検査を実施することが可能となる．

表 7 に LTL 検査項目表の例を示す．

4. 2 LTL 検査の期待結果判断

自動生成された LTL 検査項目表の期待結果欄には，LTL 検査で期待される結果を「TRUE」「FALSE」から選択する．期待結果と LTL 検査の実施結果が等しい場合に，試験結果はＯＫと判断される．

表 7　LTL 検査項目表

検証パターン	検査内容	LTL	記号定義	期待結果	実施結果	試験結果
状態到達性	「ＣＤなし」に到達するか？	!◇(s)	#define s (st == NOCD)	FALSE	FALSE	ＯＫ
状態到達性	「再生中」に到達するか？	!◇(s)	#define s (st == PLAY)	FALSE	FALSE	ＯＫ
状態到達性	「停止中」に到達するか？	!◇(s)	#define s (st == STOP)	FALSE	FALSE	ＯＫ

4. 3 LTL 検査の実行

LTL 検査支援機能では，自動生成した LTL 検査項目表に含まれる LTL 検査式に対して，バッチ処理で LTL 検査を実施することができる．これにより，LTL 検査の手順を簡略化し，大量の LTL 式を用いた検査を一度に実施することが可能となる．

LTL 検査支援機能を使用したモデル検査の流れを図 2 に示す．

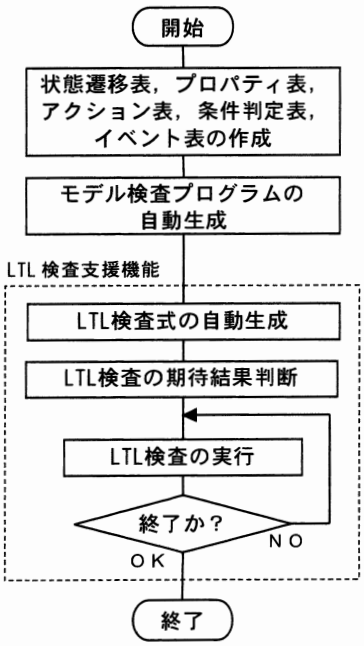

図 2　モデル検査の流れ

5.　LTL 検証パターン

LTL 検査支援機能は，あらかじめ用意された検証パターンに基づき，LTL 検査項目を自動生成する．以下に，LTL 検査支援機能が提供する LTL 検証パターンを示す．

5. 1 プロパティ到達性

プロパティ表には，システムの構成要素について，その初期値，許容される値，システムの各状態において取るべき値が定義される．それらの定義から，PROMELA プログラムに assert 文が出力される．しかし，assert 文では「取るべき値以外，取らないか？」は検証できるが，「この値を取りえるか？」などの検証はできない．

以下に，プロパティ表で定義されたシステムの構成要素について，LTL 検査支援機能で検証できる項目について示す．

5. 1. 1 許容値への到達

プロパティ表に定義した，システムの構成要素の許容される値において，実際にすべての値を取りえることを検証する．これを「許容値へのプロパティ到達性」と命名する．

許容値へのプロパティ到達性検査では，取りうるはずの値が取られない不具合を発見したり，許容値に不必要な値が列挙されているのを発見するために使用される．以下の LTL 検査式を使用する．

LTL 検査式：¬◇(p)

p：(構成要素 = 許容される値)

表 2 のプロパティ表では，以下の検査内容を含む LTL 検査項目表が生成される．

- 「ＣＤ」が「あり」になるか？
- 「ＣＤ」が「なし」になるか？
- 「モータ」が「停止」になるか？
- 「モータ」が「回転」になるか？

許容値へのプロパティ到達性の LTL 検証パターンで生成された LTL 検査式に対する期待結果は，すべて「FALSE」とする．

5. 1. 2 制約値への到達

プロパティ表に定義した，システムの各状態における構成要素の制約値の中に複数の値が存在する場合に，実際にそれらすべての値を取りえることを検証する．これを「制約値へのプロパティ到達性」と命名する．

制約値へのプロパティ到達性検査は，取りうるはずの値が取られない不具合を発見したり，制約値に不必要な値が列挙されているのを発見するために使用される．以下の LTL 検査式を使用する．

LTL 検査式：¬◇(s ∧ p ∧ lbl)

s　：(状態 = システムの各状態)
p　：(構成要素 = 取るべき値)
lbl：(PROMELA 上の検証箇所を示すラベル)

表 2 のプロパティ表では，以下の検査内容を含む LTL 検査項目表が生成される．

- 「停止中」に，「ＣＤ」が「あり」になるか？
- 「停止中」に，「ＣＤ」が「なし」になるか？

制約値へのプロパティ到達性検査の LTL 検証パターンで生成された LTL 検査式に対する期待結果は，すべて「FALSE」とする．

5. 2 状態到達性

状態遷移表でシステムをモデル化すると，システムが取りうる状態が明確になる．しかし，状態遷移表に定義された状態に実際に遷移するかどうかは PROMELA ファイルのみによるモデル検査では検証できないため，LTL による検証が必要になる．

以下に，システムの状態遷移に関して，LTL 検査支援機能で検証できる項目について示す．

5．2．1 状態への到達

状態遷移表に定義されたすべての状態に，実際に到達しうることを検証する．これを「状態到達性」と命名する．

状態到達性検査は，不具合によって到達しえない状態や，不必要な状態を発見するために使用される．以下の LTL 検査式を使用する．

LTL 検査式：¬◇(s)

s：(状態 = 到達すべき状態)

表 1 の状態遷移表では，以下の検査内容を含む LTL 検査項目表が生成される．

- 「ＣＤなし」に到達するか？
- 「再生中」に到達するか？
- 「停止中」に到達するか？

状態到達性の LTL 検証パターンで生成された LTL 検査式に対する期待結果は，意図的に到達しえない状態を作成している場合を除いてすべて「FALSE」とする．

5．2．2 状態から状態への到達

状態到達性検査において基準状態を指定し，基準状態からその他の状態への到達と，その他の状態から基準状態への到達を検証する．これを「基準状態付き状態到達性」と命名する．

基準状態付き状態到達性検査は，基準状態から到達しえない状態や，一度到達すると基準状態に戻れない状態の発見に使用される．以下の LTL 検査式を使用する．

LTL 検査式：□(s0 → □(¬s1))

s0：(状態 = 基準となる状態)
s1：(状態 = 到達すべき状態)

表 1 の状態遷移表において，基準状態を「ＣＤなし」と指定した場合，以下の検査内容を含む LTL 検査項目表が生成される．

- 「ＣＤなし」から「再生中」に到達するか？
- 「ＣＤなし」から「停止中」に到達するか？
- 「再生中」から「ＣＤなし」に到達するか？
- 「停止中」から「ＣＤなし」に到達するか？

基準状態付き状態到達性検査の LTL 検証パターンで生成された LTL 検査式に対する期待結果は，基準状態からは到達しえない状態や一度到達すると基準状態に戻れない状態を意図的に作成した場合を除いてすべて「FALSE」とする．

5．3 遷移網羅性

状態到達性検査で，状態遷移表に定義されたすべての状態に到達できることを検証しただけでは，必ずしもすべての状態遷移を網羅的に検証したとはいえない．

図 3 は表 1 の状態遷移表の状態遷移図である．「停止中」状態から「ＣＤなし」状態に到達する経路は，直接遷移する経路と「再生中」状態を経由する経路の 2 通り存在する．このように，ある状態へ到達する経路が複数存在している場合，その状態へ到達することを検証しただけでは，存在する経路をすべて通過することは検証できない．

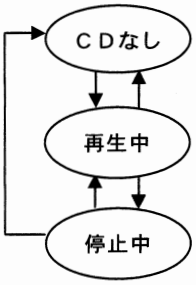

図 3　状態遷移図

もしも不具合によって状態遷移の経路が絶たれていると，絶たれている経路上に assert 文による制約違反を生じる不具合があったとしても検出することができない．そこで，状態遷移表に定義されているすべての遷移を実際に通過することを検証する必要がある．これを「遷移網羅性」と命名する．

遷移網羅性検査は，状態遷移表に定義されていながら実際には通過することのない遷移を発見するために使用される．以下の LTL 検査式を使用する．

LTL 検査式：¬◇(s ∧ e ∧ ns ∧ lbl)

s　：(状態 = 現在の状態)
e　：(イベント = 発生するイベント)
ns　：(次の状態 = 到達すべき状態)
lbl：(PROMELA 上の検証箇所を示すラベル)

表 1 の状態遷移表では，以下の検査内容を含む LTL 検査項目表が生成される．

- 「ＣＤなし」で，「ＣＤ挿入」が発生し，「再生中」に遷移するか？
- 「再生中」で，「停止ボタン」が発生し，「停止中」に遷移するか？
- 「再生中」で，「排出ボタン」が発生し，「ＣＤなし」に遷移するか？
- 「停止中」で，「再生ボタン」が発生し，「再生中」に遷移するか？
- 「停止中」で，「排出ボタン」が発生し，「ＣＤなし」に遷移するか？

遷移網羅性検査の LTL 検証パターンで生成された LTL 検査式に対する期待結果は，すべて「FALSE」とする．

5. 4 状態遷移マトリクステスト

遷移網羅性検査では，遷移のみに着目し，ガード条件の成立／不成立やアクションの実行／非実行については考慮していない．

そこで，ホワイトボックステストで使用される，C0，C1，C2 の網羅性基準を参考に，状態遷移マトリクステスト [6] における各セル内の検査の網羅性基準を策定する．

策定した網羅性基準に基づいたテストケース生成を LTL 検証パターンに追加する．

表 8 の状態遷移表を例に，状態遷移マトリクステストの実施例を示す．

表 8　状態マトリクステスト

		状態	
		状態 C	
イベント	イベント A	[条件 X] 状態 A [条件 Y] 状態 B [条件 Z] 状態 A	
		処理 a [条件 W] 処理 b	

表 8 の状態遷移表のセルは，図 4 のフローチャートで表現することができる．

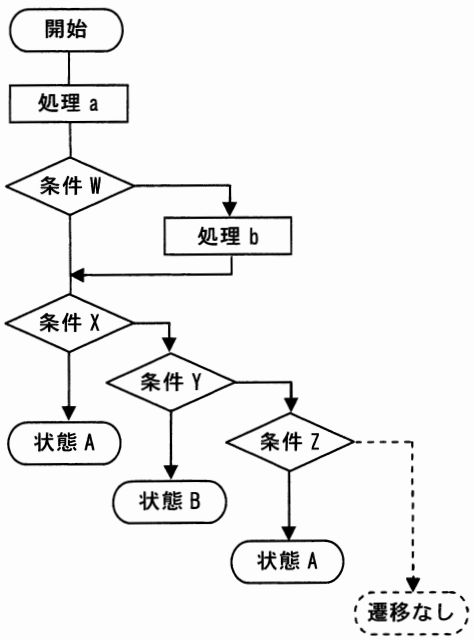

図 4　表 8 のセルのフローチャート表現

5. 4. 1 網羅性基準の定義

状態遷移マトリクステストにおける網羅性基準を，以下の M0，M1，M2 の 3 レベルで定義する．

◆ M0 レベル（命令網羅）

状態遷移表の各セルの命令だけに着目し，各々の命令が実行できることを検証する．表 8 のセルでは，命令は 2 つのアクションと 3 つの遷移であるため，以下の 5 つの検査内容を含む LTL 検査項目表が生成される．

- 「処理 a」が実行されるか？
- 「条件 W」が成立し，「処理 b」が実行されるか？
- 「条件 X」が成立し，「状態 A」に遷移するか？
- 「条件 Y」が成立し，「状態 B」に遷移するか？
- 「条件 Z」が成立し，「状態 A」に遷移するか？

◆ M1 レベル（分岐網羅）

状態遷移表の各セルの分岐と命令に着目し，各々の命令に分岐が付与されている場合，すべての分岐の方向と命令の実行性を検証する．表 8 のセルでは「処理 a」以外の命令にはガード条件が付与されているため，ガード条件が成立した場合と成立しない場合のテストケースが生成される．その結果，以下の 9 つの検査内容を含む LTL 検査項目表が生成される．

- 「処理 a」が実行されるか？
- 「条件 W」が成立し，「処理 b」が実行されるか？
- 「条件 W」が成立せず，「処理 b」が実行されないか？
- 「条件 X」が成立し，「状態 A」に遷移するか？
- 「条件 X」が成立せず，「状態 A」に遷移しないか？
- 「条件 Y」が成立し，「状態 B」に遷移するか？
- 「条件 Y」が成立せず，「状態 B」に遷移しないか？
- 「条件 Z」が成立し，「状態 A」に遷移するか？
- 「条件 Z」が成立せず，「状態 A」に遷移しないか？

◆ M2 レベル（条件網羅）

状態遷移表の各セル内の経路に着目し，通過しうるすべての経路を検証するテストケースを生成する．表 8 のセルでは，以下の 8 つの検査内容を含む LTL 検査項目表が生成される．

- 「処理 a」が実行され，「条件 W」が成立し，「処理 b」が実行され，「条件 X」が成立し，「状態 A」に遷移するか？
- 「処理 a」が実行され，「条件 W」が成立し，「処理 b」が実行され，「条件 Y」が成立し，「状態 B」に遷移するか？
- 「処理 a」が実行され，「条件 W」が成立し，「処理 b」が実行され，「条件 Z」が成立し，「状態 A」に遷移するか？
- 「処理 a」が実行され，「条件 W」が成立せず，「処理 b」が実行されず，「条件 X」が成立し，「状態 A」に遷移するか？

- 「処理 a」が実行され，「条件 W」が成立せず，「処理 b」が実行されず，「条件 Y」が成立し，「状態 B」に遷移するか？
- 「処理 a」が実行され，「条件 W」が成立せず，「処理 b」が実行されず，「条件 Z」が成立し，「状態 A」に遷移するか？
- 「処理 a」が実行され，「条件 W」が成立し，「処理 b」が実行され，「条件 X」「条件 Y」「条件 Z」が成立せず，内部遷移するか？
- 「処理 a」が実行され，「条件 W」が成立せず，「処理 b」が実行されず，「条件 X」「条件 Y」「条件 Z」が成立せず，内部遷移するか？

検証対象のセルが無視セルの場合，何も処理が実行されずに，内部遷移することを検証する．不可セルの場合は，そのセルに到達できないことを検証する．

5. 4. 2 遷移網羅性との比較

状態遷移マトリクステストでも，遷移網羅性検査と同じように，遷移先に遷移することを検証している．

しかし，遷移網羅性検査では，遷移に付与されたガード条件を考慮していない．そのため，表 8 のセルでは，「状態 A」と「状態 B」に遷移することを検証する 2 つの LTL 検査式が生成される．

それに対して，状態遷移マトリクステストでは遷移に付与されたガード条件を考慮する．表 8 のセルでは，「状態 A」に遷移するためのガード条件は「条件 X」と「条件 Z」の 2 つあるため，それぞれに対する LTL 検査式が生成され，「状態 B」に遷移する場合も含めると，3 つの LTL 検査式が生成される．

5. 4. 3 暗黙の内部遷移

状態遷移マトリクステストにおいて，M2 レベル（条件網羅）を選択した場合，表 8 では，「『条件 X』『条件 Y』『条件 Z』」が成立せず，内部遷移するか？」というテストケースを生成する．しかし，これは状態遷移表の作成者から見ればありえないケースの場合がある．もしも「条件 X」「条件 Y」「条件 Z」のいずれかが必ず成立するガード条件であったならば，前述のケースは起こりえない．

しかし，表 8 の状態遷移表の記述だけでは，「ガード条件が同時に成立するか？」や「すべてのガード条件が成立しない場合があるか？」は判断できないため，内部遷移のテストケースが生成される．このように，状態遷移表の作成者が意図しない遷移を「暗黙の内部遷移」と命名する．「暗黙の内部遷移」のテストケースの期待結果は，状態遷移表の作成者が判断しなければならない．

5. 4. 4 無視セル／不可セルの判定

モデル検査支援環境で生成した PROMELA プログラムによるモデル検査では，不可セルへの到達を assert 文で検出することができる．しかし，不可セルを誤って無視セルと記述しても，不具合として検出することはできない．

状態遷移マトリクステストで M2 レベル(条件網羅）を選択した場合，不可セルを誤って無視セルと記述すると，そのセルに到達して何も処理を実行しないことが検証されるため，到達できない不可セルであった場合には不具合として検出される．

これらによって，無視セルと不可セルを厳密に区別した場合に，その誤りをモデル検査で発見することができる．

6. プロジェクトへの適用

6. 1 適用対象

LTL 検査支援機能を含むモデル検査支援環境を，表 9 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．

表 9　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
9	車載機器の通信モジュール

6. 2 適用結果

表 9 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 10 に示す．

表 10　検出された不具合

不具合内容	件数
プロパティ制約違反	１９
アクション事前制約違反	2
状態未到達	2
プロパティ値未到達	1
計	２４

最も多く発見された不具合であるプロパティ制約違反とアクション事前制約違反は，LTL を用いず PROMELA プログラムのみによるモデル検査で発見された．プロパティ制約違反の 1 つに，遷移先のガード条件が全て成立せず，意図しなかった暗黙の内部遷移が発生することによる不具合が存在した．

LTL を用いないモデル検査で制約違反が検出さ

れなくなった後，LTL 検査支援機能を用いて LTL 検査項目を生成し，LTL を用いたモデル検査を実施した．その結果，本来到達可能でなければならない状態に到達していない不具合である「状態未到達」や，システムの構成要素が取りうるはずの値を取られない不具合である「プロパティ値未到達」が発見された．

7. 考察

7. 1 検証パターンに関する考察

状態遷移表と付加的な表から，検証パターンに基づく LTL 検査式を自動生成することにより，LTL の知識を必要とせずに，LTL 検査を実施する環境を構築した．

LTL 検査支援機能における検証パターンは，要求定義によってシステムに求められた性質というよりも，状態遷移表として表現されたシステム設計が，設計者の意図したとおりに動作することを検証するためのものである．

要求定義でシステムに求められた性質は，プロパティ表とアクション表に制約条件として記述されることを期待している．しかし，そのような性質の中にも，LTL 検査に適しているものがあるだろう．LTL の難解さをパターン化によって隠蔽しながら，LTL 検査の適用範囲を拡げることができれば，より効果的で有用なモデル検査を実施することができる．

7. 2 状態遷移マトリクステストの実用性

状態遷移マトリクステストでは，網羅性基準を選択することによって，テストケース生成のレベルを変更することができる．高い網羅性基準を選択すれば，より細かいレベルで検査することができる．しかし，高い網羅性基準を選択すると生成されるテストケースが多くなり，期待結果の判断が煩雑になる．さらに，検査時間も膨大になってしまう．

プロジェクトへの適用では，状態遷移マトリクステストにおいて不具合が検出されたという事例はない．しかし，それは状態遷移マトリクステストに不具合検出能力がないということではなく，事例では状態遷移マトリクステストの実施前に不具合が解消されていたということである．

実施のための手間を考えると，状態遷移マトリクステストは，不具合の発見よりも，状態遷移モデルが設計者の意図したとおりに動作することを最終確認し，その結果をエビデンスとして残すことに意義があると考えられる．

7. 3 モデル検査支援環境による検査の実施基準

LTL 検査支援機能を含むモデル検査支援環境を使用してモデル検査を実施する場合，以下の手順を最低限の実施基準とするのが適切であると考える．

① PROMELA ファイルのみによる検査
② ①で不具合が検出された場合，修正して①を再実施
③ 状態到達性の LTL 検査
④ ③で不具合が検出された場合，修正して①から再実施
⑤ 遷移網羅性の LTL 検査
⑥ ④で不具合が検出された場合，修正して①から再実施

③⑤の工程で不具合が検出され修正した場合，潜在していた不具合が顕在化し，assert 文により検査が中断されてしまう可能性がある．そのため，①から再実施する必要がある．

8. おわりに

本稿では，状態遷移表のモデル検査で使用される LTL 検証パターンについて考察した．検証パターンに基づく LTL 検査式の自動生成機能を含むモデル検査支援環境をプロジェクトに適用し，その有用性を評価した．

今後は，LTL 検証の適用範囲を拡大するために，LTL の難解さを隠蔽しながら，検証パターンをさらに充実させたい．

参考文献

1) G.J. Holzmann: The Spin Model Checker: Primer and Reference Manual, Addison-Wesley, 2003
2) 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, 東銀座出版社, 1998
3) 小池隆: 状態遷移表に基づくモデル検査の支援環境, 情報処理学会研究報告 Vol.2008 No.116
4) M.B. Dwyer, G.S. Avrunin and J.C. Corbett: Patterns in Property Specifications for Finite-State Verification, Proceedings of the 21st International Conference on Software Engineering, 1999
5) 金井勇人, 岸知二: UML 設計モデル検査技術のための検証パターンの提案, 情報処理学会研究報告 Vol.2006 No.48
6) 西康晴: ソフトウェアテストの概要, Open SESSAME 組込みソフトウェア技術者・管理者向けセミナー 初級者向けテキスト, 2003
7) 小池憲史, 吉田聡, 大崎人士: LTL モデル検査のための図示記法, 第 14 回ソフトウェア工学の基礎ワークショップ (FOSE2007) 予稿集