JaSST'12 Shikoku ワークショップ

Model-Based Testing入門

～グラフによる振舞いの抽象化とテストケースの設計～

2012年7月13日

高木 智彦（香川大学）

目次

ソフトウェアの仕様を表すモデルに基づいてテストケースを設計するModel-Based Testing（MBT）は古くから開発現場で利用されており，また学術的にも活発に研究が行われています．今回のワークショップでは，基本的なモデルのひとつであるグラフを用いてソフトウェアの振舞いを表し，テストケースを設計する方法について概説します．

１．MBT概論（１）

- 本ワークショップにおけるMBTの定義：
  テスト対象ソフトウェアやその運用環境の期待される振舞いを形式的モデルによって抽象化し，これに基づいてテストケースを設計，実行する方法の総称．モデルの表記法，テスト基準，テストケース設計法などからなる．特にシステムレベルのテストにおいて効果が期待できる．
- ここでいう「形式的モデル」とは，コンピュータが解釈可能な図や数式，人工言語で表されたモデルのこと．
- モデルの表記法の例としては，UMLステートマシン図，シーケンス図，FSM（Finite State Machine），マルコフ連鎖，ペトリネット，Z，CSPなどが挙げられる．他にも多数存在．

１．MBT概論（２）

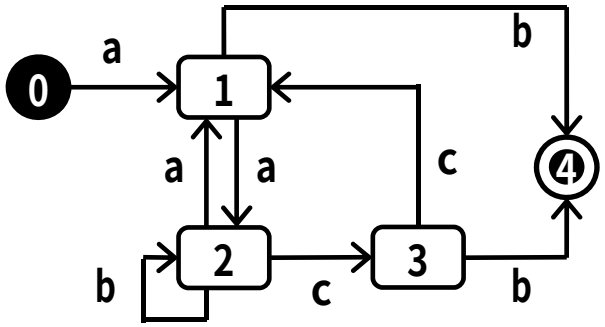
- 「抽象化」とは，「重要でない情報を省略する」ことであり，MBTにおける最も重要な概念．形式的モデルを用いて抽象化することによって，要求仕様の再確認，テストケース設計の体系化や保守性向上などが可能になる．
- モデルの表記法によって，どのように抽象化するかの大枠が決まる．さらに，モデリングの過程で意図的に抽象化を行う．モデリングは熟練を要する．
- モデルが完成すれば，テストケースの設計をテスト基準にしたがって自動的に行うことができる．テスト基準は，モデルの構造に着目したもの（たとえば状態網羅），統計的性質に着目したもの（たとえばMTTF）などがある．

１．MBT概論（３）

- テストケースの設計には，グラフ探索アルゴリズムや記号実行，ランダムウォークなどが用いられる．
- MBTのテストケースは抽象化されているので，テスト対象ソフトウェアとのギャップがある．よって，テストケースを実行するには，テストケースを具象化するか，テスト対象ソフトウェアの実行結果を抽象化しなければならない．自動実行にはツールが必要．
- テストケースを実行した結果，もしフォールトが発見されればそれを除去することによってテスト対象ソフトウェアの信頼性を改善できるし，発見されなければテスト対象ソフトウェアの信頼性について確信を深めることができる．

２．グラフによる抽象化（１）

- グラフ（Graph）は，ノード（Node）とアーク（Arc）から構成される単純なモデルであり，MBTで用いられる多くのモデルの基礎となっている．
- 本ワークショップでは，テスト対象ソフトウェアの状態をノード，遷移をアーク，遷移を引き起こすイベントを当該アークのラベルとしてテスト対象ソフトウェアの振舞いをモデリングする．



	a	b	c
0（開始状態）	1		
1	2	4	
2	1	2	3
3		4	1
4（終了状態）			

図１．グラフの簡単な例（図と表による表現）

２．グラフによる抽象化（２）

- 状態は，テスト対象ソフトウェアの振舞いを特徴付ける変数（状態変数）の値によって識別する．状態変数は，プログラムの変数を抽象化したものだったり，プログラムにおける処理の段階を抽象化したものだったりする．
- イベントは，状態変数の値を変化させる操作である．プログラムにおける関数呼出しを抽象化したものだったり，外部システムやユーザからの入力を抽象化したものだったりする．テスト対象ソフトウェア自身がイベントを生成することもある．

２．グラフによる抽象化（３）

- テスト対象ソフトウェアの振舞いを表すグラフは，通常以下の条件を満たしていなければならない．
  1. すべての状態が開始状態から到達可能であること．
  2. 終了状態が存在する場合，すべての状態が終了状態に到達可能であること．
  3. すべての状態において，生起するイベントが与えられれば，遷移先が一意に定まること．

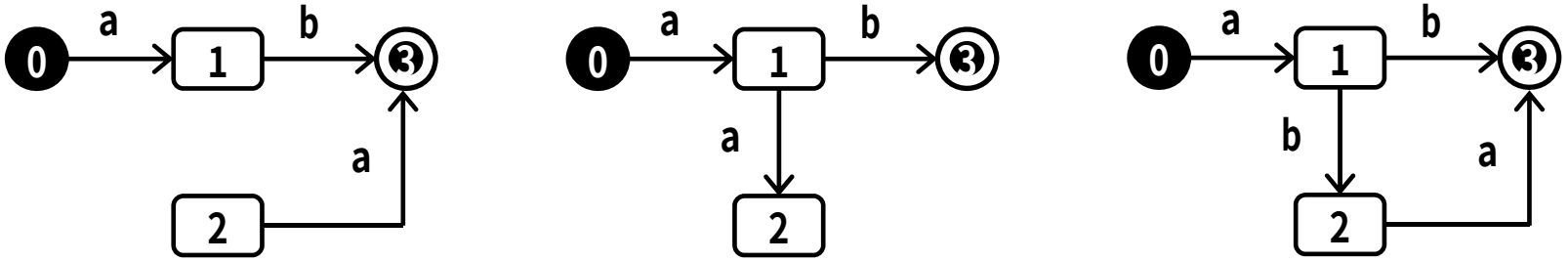


図２．誤ったモデルの例
（左図：1を満たさない，中図：2を満たさない，右図：3を満たさない）

２．グラフによる抽象化（４）

- テスト対象ソフトウェアを構成する個々のサブシステムやモジュールの振舞いを表すグラフを作成し，これらを合成することでテスト対象ソフトウェアのモデルを導出してもよい．

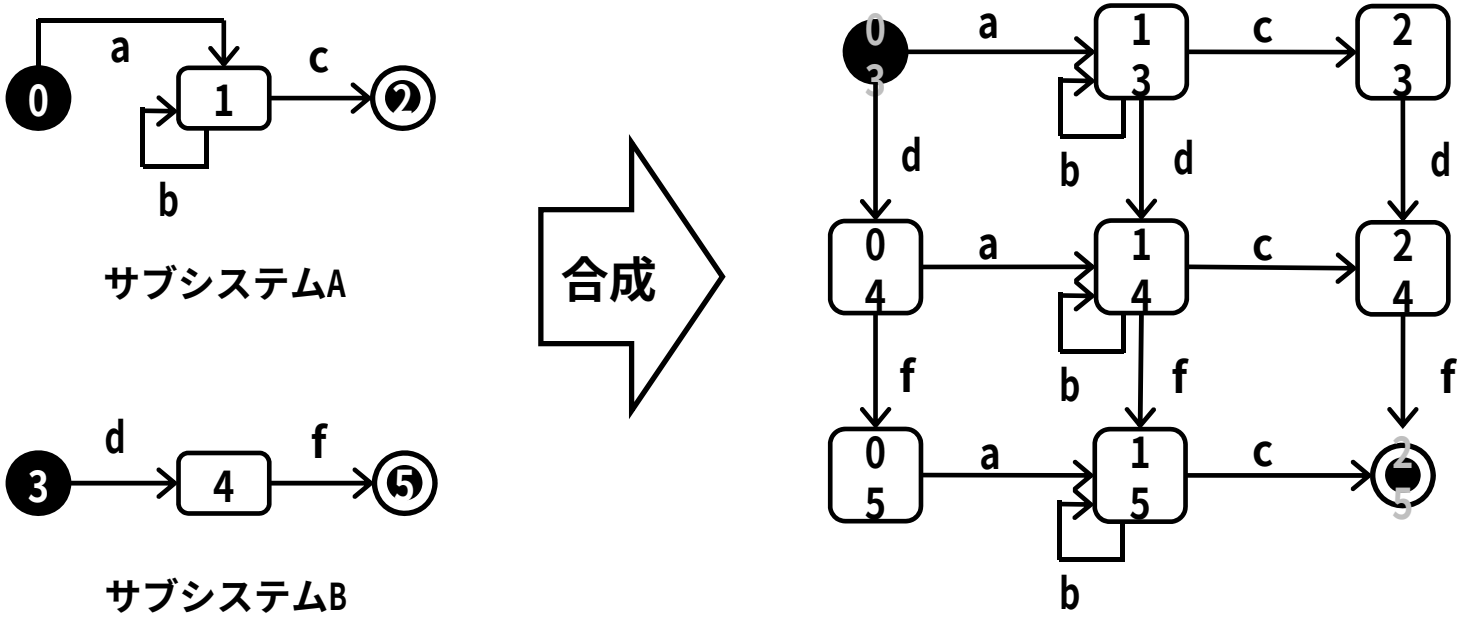


図３．サブシステムAとBの合成例

２．グラフによる抽象化（５）

＜演習１＞ 基本問題

あるテレビ番組録画再生ソフトウェアは録画機能と再生機能を同時に利用できる仕様になっている．録画機能と再生機能がそれぞれ下図のように定義された場合，録画機能と再生機能の合成された振舞いを表すグラフを定義せよ．

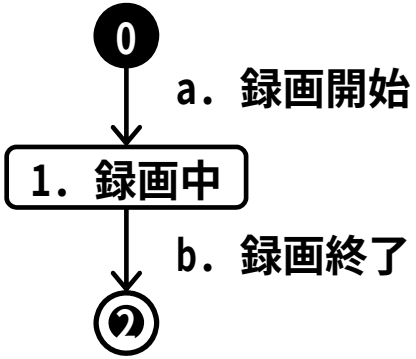


図４．録画機能のグラフ

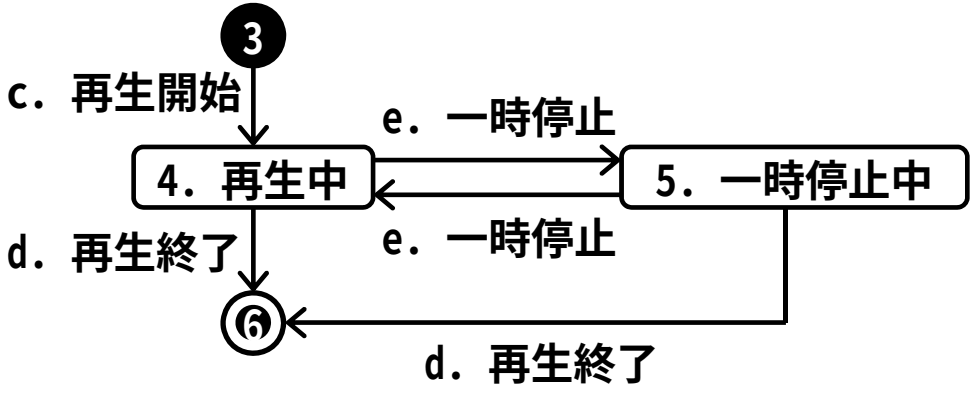


図５．再生機能のグラフ

２．グラフによる抽象化（６）

＜演習２＞ 基本問題

以下に示す携帯電話用メールソフトウェアのメッセージ新規作成機能の振舞いをグラフで定義せよ．時間が余った場合は，抽象度の異なるものをもうひとつ定義せよ．

- 「メッセージ新規作成」ボタンを押下すると，当該機能のメイン画面（図６）が呼び出される．呼び出し直後のタイトル，送信先アドレス，本文の各フィールドは空欄である．「キャンセル」ボタンを押下すると，当該機能は終了する．
- 「タイトル編集」「アドレス編集」「本文編集」のいずれかのボタンを押下すると，対応するフィールドのテキストを編集する画面（図７）に移行する．「決定」または「キャンセル」ボタンを押下すると，テキスト編集を終了しメイン画面に戻る．「決定」は編集内容を対応するフィールドに反映させるが，「キャンセル」は反映させない．なお，各フィールドは任意の順番で繰り返し編集できる．
- アドレスのフィールドが空欄でなければ，「送信」ボタンを押下してメッセージを送信することができる．送信完了後，当該機能は自動的に終了する．
- その他，必要に応じて各自で自由に仕様を仮定してよい．

２．グラフによる抽象化（７）

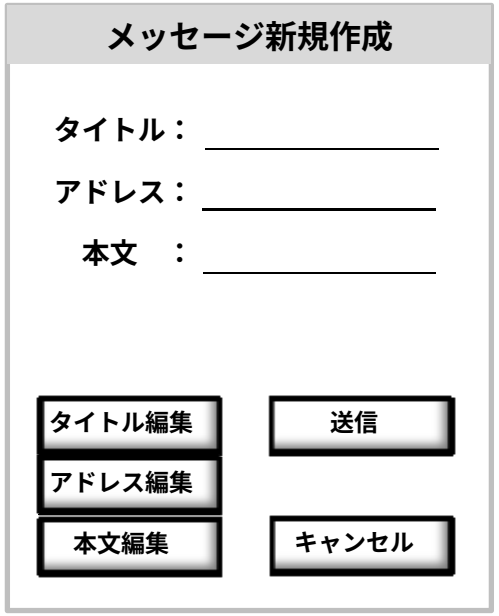


図６．メイン画面

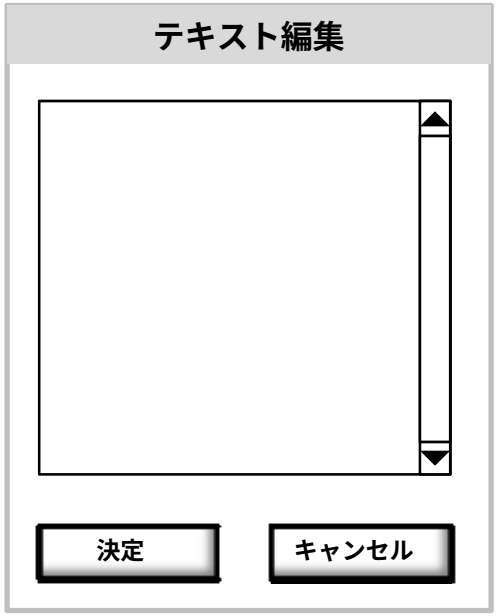


図７．テキスト編集画面

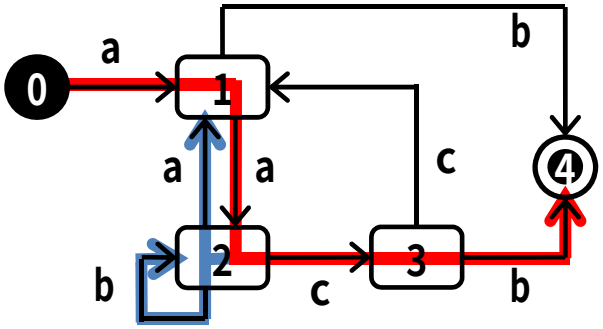
３．グラフに基づくテストケース設計（１）

- グラフから生成されるテストケースは，開始状態で始まる連続する遷移の列である．テスト基準を満たすには，通常は複数のテストケースが必要である．
- テスト基準を満たすテストケースをグラフ探索アルゴリズムによって自動生成することができる．
- 最も一般的なテスト基準として，すべての状態を少なくとも１回実行する状態網羅基準がある．グラフ探索アルゴリズムのひとつである深さ優先探索※によって，これを満たすテストケースを容易に生成できる．

※　深さ優先探索で生成されるテストケースは冗長になりがちである．より少ないテストケースを生成するアルゴリズムとしてChinese Postman Algorithmが知られている．

３．グラフに基づくテストケース設計（２）

（１）　開始状態を現在の状態とする．
（２）　現在の状態を遷移元とする任意の遷移を選択する．一度選択した遷移には印を付けておき，以降の選択対象から除外する．そして，もし遷移先の状態が未実行であれば当該遷移を実行し現在の状態を更新する．以上を，現在の状態において選択可能な遷移がなくなるまで繰り返す．
（３）　もし，現在の状態がそれまで出力したテストケースに含まれていなければ，探索経路（開始状態から現在の状態に至る遷移列）をテストケースとして出力する．図８参照．
（４）　その遷移列をさかのぼり，選択可能な遷移が残っている直近の状態を現在の状態とし（２）に戻る．そのような状態がない場合は探索を終了する．



（２）において選択可能な遷移が複数ある場合アルファベット順に選択すると仮定すると，赤でハイライトされた遷移列がテストケースとして出力される．なお，青でハイライトされた部分は，テストケースには含まれなかった探索済みの部分である．

図８．深さ優先探索による状態網羅基準を満たすテストケースの生成過程

３．グラフに基づくテストケース設計（３）

＜演習３＞ 基本問題

演習２で作成したグラフから，状態網羅基準を満たすテストケースを設計せよ．また，抽象度の違いによってテストできる内容にどのような差が生じるか考察せよ．

３．グラフに基づくテストケース設計（４）

＜演習４＞ 基本問題

１４ページのアルゴリズムを少し変更すれば，遷移網羅基準を満たすテストケースを生成できる．演習２で作成したグラフから遷移網羅基準を満たすテストケースを設計せよ．

３．グラフに基づくテストケース設計（５）

＜演習５＞ 発展問題

モデルが誤っていると適切なテストケースを得ることができない．８ページに示した条件はどのような方法で検証することができるか考察せよ．

３．グラフに基づくテストケース設計（６）

＜演習６＞　発展問題

**　Nスイッチ網羅基準を満たすテストケースを演習２で作成したグラフから設計せよ．**

４．まとめ

- MBTで利用されるモデルやテスト基準には数多くの種類があり，それぞれについて適切なテストケースを生成する方法が存在する．
- MBTの成否の鍵はモデリングである．モデリングで重要なのは仕様の抽象化であり，ある程度熟練を要する．
- MBTではテストケースを自動生成することができる．ただし，テストケースの実行を自動化するにはそれなりの仕掛けが必要である．
- MDDツールがモデルの表記法やコード生成法を拡張する機能を提供するように，将来MBTツールもモデルの表記法やテストケース設計法などを拡張できるようになるかもしれない．