研究コース５（チーム形式仕様）

テストケース自動生成における
テスト仕様検証への形式手法適用

Formal Method for Test Specification in Automated Test Case Generation

研 究 員　　：井森　一樹（富士通株式会社）
　　柳田　憲彦（エヌ・ティ・ティ・コムウェア株式会社）
主 査　　：栗田　太郎（ソニー株式会社）
副 主 査　　：石川　冬樹（国立情報学研究所）
アドバイザー：荒木　啓二郎（熊本高等専門学校）

研究概要
テスト仕様を入力とし，組み合わせの技法を使って自動でテストケースを生成するアプローチでは，人が作成するテスト仕様に誤りが混入する．
テスト仕様の誤り修正において，全テストケースとテスト仕様の対応確認は困難な作業であり，テスト仕様に対する検証と誤り箇所特定が課題となっている．
本研究では，形式手法により正確かつ簡易に課題を解決する，「FMTS 法(Formal Method for Test Specification)」を提案する．
形式手法に Alloy を採用した実装例の評価では，テスト仕様に対し正確な検証と誤り原因特定検証が可能であることがわかり，困難であったテスト仕様の修正が可能となった．

1. はじめに

テスト仕様を入力とした，テストケース自動生成には，入力値生成と期待結果生成の技術領域がある．このうち期待結果生成は更なる研究が必要な分野である．テスト仕様は人が作成するため，誤りが混入する．しかし，この誤りの原因特定や検証に対する研究は進んでいない．

誤りの中には，テスト仕様上の誤り箇所が不明のため，修正が困難なものがある．この誤り箇所を特定するには，全テストケースとテスト仕様を突き合わせる必要がある．組み合わせの技法を使用してテストケースを自動生成する場合,テストケース数が膨大となり，現実的に突き合わせによる誤り箇所特定はできない．テスト仕様の誤りを修正するためには，テスト仕様の検証と誤り箇所の特定が課題である．

本研究では，この課題の解決策として FMTS 法を提案する．FMTS 法は，その実装を簡易にするため，形式手法を適用した．

FMTS 法の実装では，FMTS 法の機能と相性がよい Alloy を，形式手法として採用した．

実装例の評価では，誤りが有る場合と無い場合の評価データを用意し，誤りの有無を検証できるか，誤り箇所を特定できるかを確認した．評価の結果，FMTS 法の実装例が，誤りの有無の検証や，誤り箇所特定が実施できることを確認できた．

組み合わせの技法を使ったテストケースの自動生成では，形式手法を利用して，テスト仕様を検証し誤り箇所特定することで，テスト仕様の修正が可能であることがわかった．

以下，2 章では課題設定について述べる．3 章では解決策を提案する．4 章では解決策の評価について述べる.5 章では本研究の結論を述べる.6 章では今後の発展について述べる．

2. 課題設定

2.1　現状分析

組み合わせの技法を利用したテストケース自動生成における，テスト仕様の誤り修正の難しさを，本研究が取り組むべき問題として選択した．選択に至った現状分析を以下に記載する．

テストケースの自動生成では，テスト仕様が生成ツールへの入力となる．テスト仕様は，入力値，適用則，期待結果からなり，人がテスト対象のドキュメントを元に作成する．テストケースは，入力値と期待結果からなり，生成ツールがテスト仕様を元に生成する．

生成ツールの処理は，技術領域により入力値生成と期待結果生成に分類される[1]．入力値生成は，研究が進み[1]，代表的な手法には組み合わせの技法[2]がある．入力値生成は，テスト仕様の入力値からテストケースの入力値を出力する．期待結果生成は，更なる研究が必要な領域[3][4]であり，入力値生成と比較し扱う情報が多い．期待結果生成では，テスト仕様の適用規則・期待結果やテストケースの入力値を元に，テストケースの期待結果を出力する（図 1)．テスト仕様，テストケースの情報は(表 1)を参照．

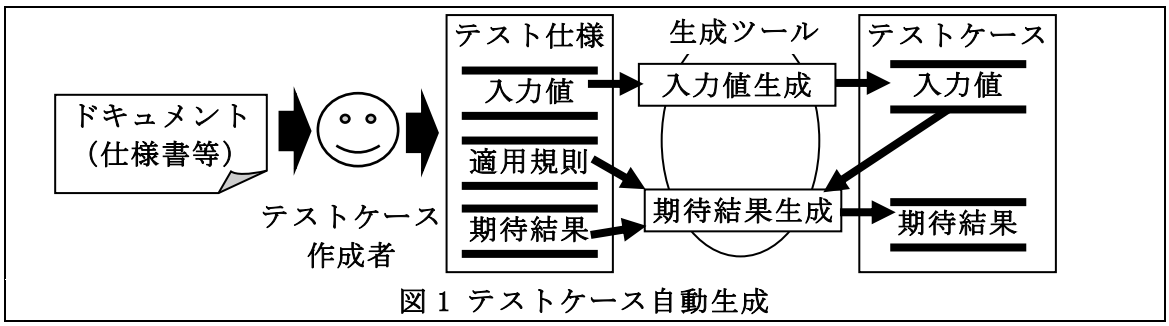

図 1 テストケース自動生成

表 1 テスト仕様・テストケースの情報

入力値	因子	例：コマンドのパラメタ
	水準	例：パラメタの値
期待結果	出力	例：終了コード，標準出力，標準エラー出力
	期待値	例：終了コードの具体的な値（0，-1 等）
適用規則		期待値を適用する条件を，因子と水準で表現

テスト仕様は，人が作成するため誤りを含む．特に，期待結果生成において，テスト仕様の適用規則，期待結果の誤りを作り込み易く，テストケースの期待結果の誤りにつながる．誤りを作り込み易い原因は，期待結果生成は研究が進んでいないこと，および，扱う情報が多いことにある．

テスト仕様の誤りには,修正が容易な誤りと困難な誤りがある.作り込み易い適用規則，期待結果誤りの例を以降に記載する．なお，記載簡略化のため，本資料内での期待結果の表記では，出力は 1 種類とし記載を省略し，列ラベルは期待値の記載を省略し，期待結果と記載する．

修正が容易な例 1 の誤りを(表 2)(表 3)に示す．例 1 では，テスト仕様上の重複や漏れの誤り箇所がわかるため，修正し易い．

表 2 例 1:適用規則・期待結果

適用規則			期待結果	誤り
A	B	C		
A1	−	−	R1	重複
A1	−	−	R2	
A2	−	−	−	漏れ

表 3 例 1: テストケース

No	入力値			期待結果	誤り
	A	B	C		
1	A1	B1	C1	R1,R2	重複
:	:	:	:	:	:
5	A2	B1	C1		漏れ
:	:	:	:	:	:

修正が困難な誤り例 2 を(表 3) (表 4)に示す．この例では，テストケース上の期待結果の誤りとして,重複や漏れが分かるにも関わらず,テスト仕様上の誤り箇所不明が原因で,修正が困難になっている．

表 3 例 2:適用規則・期待結果

適用規則			期待結果	誤り
A	B	C		
A1	−	−	R1	不明
A2	B1	−	R2	不明
−	B2−	C1	R3	不明

表 4 例 2: テストケース

No	入力値			期待結果	誤り
	A	B	C		
:	:	:	:	:	:
3	A1	B2	C1	R1,R3	重複
:	:	:	:	:	:
8	A2	B2	C2		漏れ

例 2 のように，テストケースの期待結果に誤りがあるが，テスト仕様上の誤り箇所が不明な場合，全てのテストケースとテスト仕様を突き合わせて，テスト仕様上の誤り箇所を特定する必要がある(表 5) (表 6)．

表 5 誤り箇所特定:テスト仕様

適用規則			期待結果	誤り
A	B	C		
A1	−	−	R1	不明
A2	B1	−	R2	不明
−	B2−	C1	R3	不明

表 6 誤り箇所特定:テストケース

No	入力値			期待結果	誤り
	A	B	C		
1	A1	B1	C1	R1	
2	A1	B1	C2	R1	
3	A1	B2	C1	R1,R3	重複
4	A1	B2	C2	R1	
5	A2	B1	C1	R2	
6	A2	B1	C2	R2	
7	A2	B2	C1	R3	
8	A2	B2	C2	—	漏れ

組み合わせの技法を利用してテストケースを自動生成する場合，テストケース数が膨大になる．このため，人が全テストケースとテストケースを突き合わせ，誤り箇所を特定することが難しく，テストケースの期待結果に誤りに気付いても，テスト仕様の適用規則・期待結果の誤りを修正するのは困難である．

2.2 課題提起

本研究では，テスト仕様に対する検証と誤り箇所特定を課題として設定する．以下にその理由を記載する．

2.1 では，組み合わせの技法を利用したテストケース自動生成における，テスト仕様の誤り修正の難しさを取り組むべき問題として選択した．修正が難しい原因は，例 2 で述べたように，テスト仕様の適用規則や期待結果から，誤り有無や，誤り箇所が分からない事にある．この原因の解消には，テスト仕様に対する，誤り有無の検証と，誤り箇所の特定が必要である．

2.3 先行研究

Prowell らはシーケンスベースドソフトウェア仕様記述法の研究[5]において，入力値，期待結果，適用規則での仕様記述法を提案している．例えば，ドアの安全制御システムでは，連続する操作を入力値，操作のレスポンスを期待結果，レスポンスの条件を操作の適用規則としている．しかし，この研究では適用規則や期待結果の誤り・検証については言及していない．

Balcer らの研究[6]では，入力値，および，適用規則と期待結果を人が作成し，それを元にテストスクリプトを自動生成する手法が提案されている．しかし，この研究では，適用規則と期待結果の誤り・検証については言及していない．

林祥一は，組み合わせテストに対して形式仕様記述言語（Alloy）を適用する例[7]を提示している．この例では，ツールが生成するテストケースに誤りがないかを，利用者が作成したテスト対象固有のルール（禁則）を使って検証している．しかし，利用者が作成したテスト対象固有のルール（適用規則や期待結果）自体の誤りや検証については言及してない．

2.4 Alloy とは

今回の研究で利用した Alloy は，モデルのクラス（型構造）や制約を与えると，それらを満たすインスタンスを網羅的に検証する，形式仕様記述言語のひとつであり，以下の特徴[8][9][10]がある．

(1) オブジェクト指向モデリングの影響を受け，データ間の関連をプロパティにより関連付けしやすい言語を持つ
(2) 強力な分析・検証ツールを持ち，作成中モデルを即座に検証し結果を出力する
(3) モデル検査法により，誤り箇所を特定する情報を持つ反例を発見する
(4) データを，GUI や XML 形式で整形し，見やすくて表示する

3. 解決策の提案

3.1 FMTS 法の概要

課題である，テスト仕様に対する検証と誤り箇所特定を，形式手法により解決する FMTS 法[図 2]を提案する．

FMTS 法の利用者はテストケース作成者である．利用者はテスト仕様を FMTS 法に入力する．

FMTS 法はテスト仕様を検証，検証結果を出力する．検証結果は，テストケースの期待結果誤りにつながる，テスト仕様上の適用規則の誤りの有無である．検証する誤りは適用規則の重複と漏れである．また，FMTS 法はテスト仕様の誤り箇所を特定し，誤り箇所を出力する．重複の誤りの場合，重複した適用規則を誤り箇所として出力する．漏れの誤りの場合，漏れたテストケースと漏れに関連する適用規則を，誤り箇所として出力する．

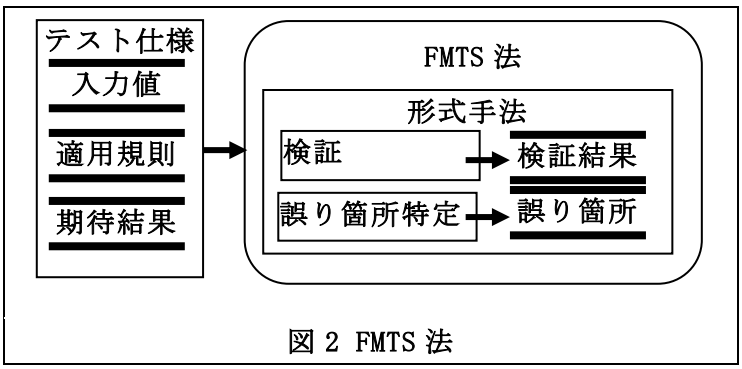

図 2 FMTS 法

テストケース利用者は出力された検証結果と誤り箇所を元に，テスト仕様の誤りを修正する．

FMTS 法では形式手法により実装を簡易にする．形式手法は，形式仕様言語，形式検証の方法，支援ツールから構成される[10]．この構成を利用し，FMTS の機能を実装する(表 7)．

表 7 形式手法の利用

形式手法の構成(特徴)	FMTS 法の機能
形式仕様言語(厳密な仕様定義)	テスト仕様定義
形式検証の方法(自動検証)	検証，誤り箇所特定
支援ツール(デバッグ用表示)	検証結果表示，誤り箇所表示

3.2 FMTS 法の実装

本研究では，FMTS 法により課題が解決できるかを評価するため，形式手法である Alloy を採用し,FMTS 法の実装例を作成した.Alloy の特徴は FMTS 法の機能と相性が良く(表 8)，FMTS 法を簡易に実装できると見込み，Alloy を採用した．

表 8 Alloy の特徴と FMTS 法の機能

Alloy の特徴(機能)	FMTS 法の機能
データを関連付けしやすい言語仕様(Alloy 言語)	テスト仕様定義
即座に検証結果を取得(Alloy 解析器)	検証，誤り箇所特定
反例情報から誤り箇所の情報を取得(モデル検証)	誤り箇所表示
GUI/XML で見やすく整形された情報(ビューア)	検証結果表示，誤り箇所表示

FMTS 法実装例の実行手順を以下に記載する．(1)の作業は，FMTS 法実装例の開発者が数分で実施した．

(1) 利用者が，テスト仕様を Alloy 言語として定義
(2) FMTS 法実装例が，テスト仕様に対し，検証と誤り箇所特定を実施
(3) FMTS 法実装例が，検証結果と誤り箇所を出力
(4) 利用者が，検証結果と誤り箇所を参照し，テスト仕様を修正

FMTS 法実装例の構成[図 3]を以下に記載する．

1)Alloy 言語は，クラス，インスタンス，制約で構成される．クラスでは型を定義する．インスタンスではテスト仕様を定義する．制約では検証用データ，重複検証ルール，漏れ検証ルールを定義する．クラスと制約は汎用性があり，対象となるテストに影響を受けないため，FMTS 法から実装済の Alloy 言語ソースを提供する．インスタンスはテスト対象によって異なるため，利用者が，テスト仕様を Alloy 言語として定義する．Alloy 言語ソースは以下の 5 ファイルで構成される．Alloy 言語ソースの詳細は（付録 1）を参照．

(1) class.als　　型定義
(2) ins.als　　テスト仕様定義
(3) generation.als　検証用データ定義
(4) dupchk.als　　重複検証ルール
(5) leakchk.als　　漏れ検証ルール

2)Alloy 解析器は，検証・誤り箇所特定を実行する．3)メッセージ・ビューアは，検証結果・誤り箇所を表示・出力する．2)Alloy 解析器，3) メッセージ・ビューアは開発・修正が不要であり，Alloy が提供する機能を，そのまま FMTS 法の実装として利用できる．

簡易な Alloy での実装の例を示す．特に，検証ルール定義は数ステップ[図 4] [図 5]と簡易であった.今回は Alloy 言語の学習と並行し,1.5 か月でて FMTS 法実装例を開発した．

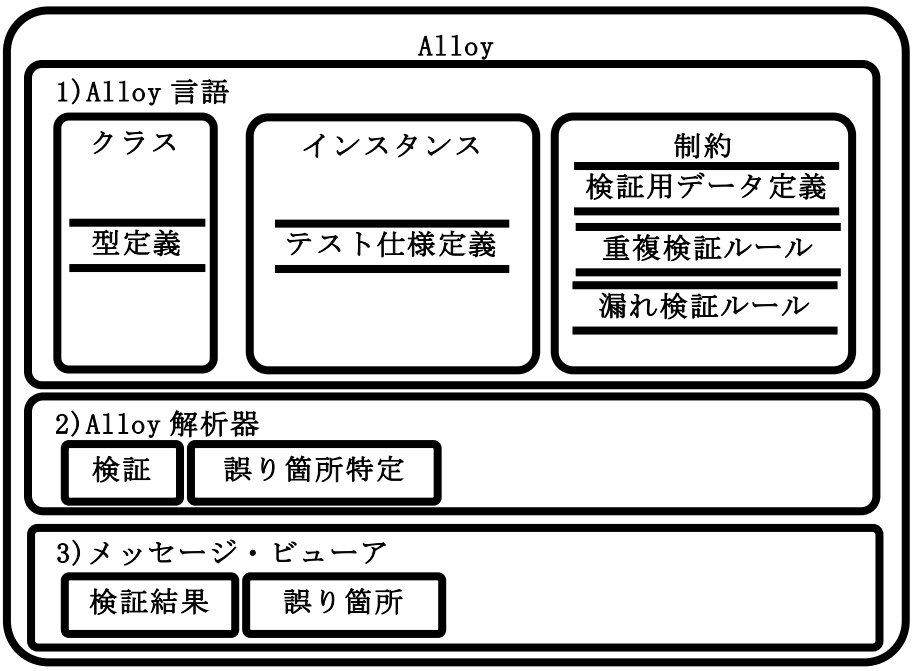

図 3 FMTS 法実装例

```
assert assert_pred_duppair {
    pred_duppair [DupPair]}
pred pred_duppair [dp :DupPair ] {
    dp.dupAep.exorder != dp.dupBep.exorder or //1.コマンドが異なる
    dp.dupAep.tyname  != dp.dupBep.tyname or  //2.出力が異なる
    dp.dupAep.exval   =  dp.dupBep.exval or   //3.期待値が同じ
    all pcic_a : dp.dupAep.pc_ic |
         no pcic_b : dp.dupBep.pc_ic |       //4.テストケースが異なる
            pcic_a = pcic_b }
```

図 4 重複検証ルール(dupchk.als より抜粋)

```
assert assert_pred_leakone {
         pred_leakone[Leaks] }
pred pred_leakone[lk : Leaks]{
    InputCombi = lk.leak_ic}    //全テストケース InputCombi と比較
```

図 5 漏れ検証ルール(leakchk.als より抜粋)

4. 解決策の評価

4.1 評価方法

課題である，テスト仕様に対する検証と誤り箇所特定を，形式手法によって FMTS 法が解決できるかを評価した．評価では，テストケースの期待結果誤りにつながる，テスト仕様上の適用規則に着目し，課題を分解した以下の RQ(Research Question)が達成されたを FMTS 法実装例を使って確認した．

［テスト仕様の検証］

(RQ1) 適用規則に重複の誤りが有ることを，検証結果に出力できるか?

(RQ2) 適用規則に漏れの誤りが有ることを，検証結果に出力できるか?

(RQ3) 適用規則に重複の誤りが無いことを，検証結果に出力できるか?
(RQ4) 適用規則に漏れの誤りが無いことを，検証結果に出力できるか?
[テスト仕様の誤り箇所特定]
(RQ5) 適用規則に重複の誤りが有る場合，重複した適用規則を出力できるか?
(RQ6) 適用規則に漏れの誤りが有る場合，漏れたテストケース，および，
漏れに関連する適用規則を出力できるか?

評価は以下の手順で実施した.

(1) 評価用データとして，テスト仕様である入力値，適用規則，期待結果，および，正解データであるテストケースを用意する（付録 2).
(2) テスト仕様を，Alloy 言語として ins.als に定義する
(3) 重複の検証，および，誤り箇所箇所特定のため，Alloy 解析器から dupchk.als を実行する
(4) 漏れの検証，および，誤り箇所箇所特定のため，Alloy 解析器から leakchk.als を実行する
(5)(3),(4)の出力を，Alloy のメッセージ，ビューアにより確認し，正解データと比較する

4.2 評価結果

評価結果を(表 9)に示す．評価結果詳細は（付録 3）を参照.

表 9 評価結果

No	評価結果	判定
RQ1	適用規則に重複の誤りが有ることを，検証結果に出力できた	○
RQ2	適用規則に漏れの誤りが有ることを，検証結果に出力できた	○
RQ3	適用規則に重複の誤りが無いことを，検証結果に出力できた	○
RQ4	適用規則に漏れの誤りが無いことを，検証結果に出力できた	○
RQ5	適用規則に重複の誤りが有る場合，重複した適用規則を出力できた	○
RQ6	適用規則に漏れの誤りが有る場合，漏れたテストケース，および，漏れに関連する適用規則を出力できた	○

4.3 評価結果の考察

形式手法により,テスト仕様に対する検証と誤り箇所特定を実現できることがわかった.

5. 結論

形式手法によるテスト仕様の検証・誤り箇所検証により，組み合わせの技法を使ったテストケース自動生成における，適用規則・期待結果の誤りを容易に修正できることがわかった.

今回の研究では，形式仕様記述中の誤り原因を特定しているが，形式仕様記述に馴染みが無い利用者にとっては利便性向上が今後の残課題である．このため，利用者が馴染みある形式で作成した期待結果データに対し，誤り原因を提示する解決案を検討していく.

6. 今後の発展

今回の研究での残課題は，性能効率性の測定と，大規模なテスト仕様での実用性の測定である．今回は，評価データとして 3 因子×2 水準の入力値を使い，テストケースは 8 個であった．テスト入力値，テストケース数を拡大し，今回の FMTS 実装例に対する評価を継続する.

研究を次の段階に発展させるためには，以下の 2 つの新たな課題があると考える．こん後はこの新課題に取り組んでいく.

(1) テスト仕様の Alloy 言語定義作業不要化
今回の研究では，Alloy 解析器やビューアなど Alloy の機能をそのまま利用することで，FMTS 法の実装を容易にした．また，FMTS 法として実装済の Alloy ソースを提供し，利用者負担を減らした．しかし，テスト仕様を Alloy 言語で定義する作業は利用者が実施する必要がある．この定義作業は，Alloy の知識が無い人にとって敷居が高い．この定義作業の不要化が新課題である．この課題の解決策として，利用者になじみある形式（Excel 等）のテスト仕様を，Alloy 言語に変換する手法などを検討する．

(2) Alloy 言語化前のテスト仕様上の誤り箇所特定
今回の実装例では，Alloy 言語化したテスト仕様定義上の誤り箇所特定ができた．利用者の利便性を高めるため，Alloy 言語化前の形式のテスト仕様に対し．誤り箇所を特定することが新課題である．Alloy 言語化前の形式は(1)の新課題と一緒に検討していく．

謝辞

本研究においては，研究会主査・副主査の栗田氏・石川両氏より，例会，特別講義などで研究の進め方や形式仕様を含む技術面で多大なご指導をいただいたことを深く感謝する．熊本高等専門学校 荒木啓二郎氏，南山大学 張漢明氏からは研究に対し有益なご助言をいただいたことを深く感謝する．

参考文献

[1] 丹野治門，テスト自動化の基本的な考え方と NTT における研究開発の紹介，第 25 回 SPI トワイライトフォーラム，2019
[2] SQuBOK 策定部会編，ソフトウェア品質知識体系ガイド（第 2 版）－SQuBOK Guide V2－，オーム社，2014
[3] Yogesh Singh, Automated Expected Output Generation-Is this a Problem that has been solved?, ACM SIGSOFT Software Engineering Notes, v.40 n.6, 2015
[4] E.T. Barr, M. Harman, P. McMinn, M. Shahbaz, S. Yoo, The Oracle Problem in Software Testing: A Survey, IEEE Transactions on Software Engineering, vol 41 no.5, 507-525, 2015
[5] S. J. Prowell and J. H. Poore , Foundations of sequence-based software specification, IEEE Transactions on Software Engineering, vol. 29 no. 5, 417-429, 2003
[6] M. Balcer, W. Hasling, and T. Ostrand, Automatic generation of test scripts fromformal test specifications, ACM SIGSOFT Software Engineering Notes, v148, 210-218, 1989
[7] 林祥一，https://qiita.com/sho1884/items/283f55c4e7e1ae374d74, (2020 年 1 月 29 日参照)
[8] 石川冬樹，「形式手法と仕様記述」の探求に向けて～形式手法とは?, 日本科学技術連盟 SQiP 研究会「形式仕様と仕様記述」, 2012
[9] D.Jackson, 抽象によるソフトウェア設計:Alloy ではじめる形式手法, 中島震(監訳), オーム社, 2011
[10] 中島震，モデル検査法のソフトウェアデザイン検証への応用，コンピュータ ソフトウェア，23 巻，2 号，p. 2_72-2_86 ,2006
[11] 中島震，形式手法入門ーロジックによるソフトウェア設計，オーム社，2012