1M2-OS-02b-4

解集合プログラミングによるソフトウェアテストケースの生成

Generating software test cases with Answer Set Programming

小俣仁美 *1　井上克巳 *1*2
Hitomi Komata　Katsumi Inoue

*1東京工業大学
Tokyo Institute of Technology

*2国立情報学研究所
National Institute of Infomatics

Developers test their software to find bugs in its program. They generate the good set of test cases that fild more bugs but it is hard to get the best set manually. This paper presents a generating test cases technique with Answer Set Programming to write objects and conditions.

1. はじめに

ソフトウェア開発において，プログラムが設計の意図の通りに動作し，意図しない動作を行わないことを検証するためには，ソフトウェアテストによりプログラムの誤りを発見することが重要である．

ソフトウェアテストの手法の種類の代表的なものとして，ホワイトボックステストとブラックボックステストが知られている [Myers 06]．ホワイトボックステストとはプログラム内部の構造についてテストする方法であり，論理構造を元にしてテストケースを作成する．具体的には，条件分岐や繰り返し処理のようなプログラムのフローに着目した手法が含まれる．一方，ブラックボックステストは内部構造について関知せず，ソフトウェアの仕様書を元にテストケースを生成する方法である．入力データに着目した手法であり，入力データの種類や値の変域に着目してテストケースの数を減らす手法が含まれる [Neate 06][Khan 12]．

ホワイトボックステスト及びブラックボックステストのどちらの方法においても，プログラムに含まれる全ての誤りを発見するには，可能な入力を全て試さなければならない．しかし，そのような入力は天文学的な数に及ぶため，全ての入力を用いてテストを行うことはできない．したがって，テストケースの生成においては，有限個数のテストでありながら発見される誤りの数がなるべく多くなるようなテストケースの集合を求めることを求められる．

提案手法では，解集合プログラミングを用いることで，ブラックボックステストの観点及び入力データに課される条件をもとに，対象ソフトウェアに必要となるテストケースを自動的に生成する．解集合プログラミングとは複雑な探索問題や組み合わせ問題に対する宣言型のプログラミングである [Lifschitz 08]．解集合プログラミングを用いることにより入力データのパラメータとその条件の表現が容易となり，複数の入力データからなる解集合としてテストケース群を導出することができる．

自動化した提案手法を Myers の三角形テスト [Myers 06] に適用して評価を行い，生成したテストケースがテストすべき必要なパラメータの組み合わせを含むことを確認した．

本稿の構成を以下に示す．2. 章においてソフトウェアテストの，3. 章において解集合プログラミングの特徴の説明を行う．4. 章で提案手法について説明し，5. 章で提案手法の評価について述べる．6. 章においてテストケース生成に関連する手法について述べる．最後に，7. 章で本稿をまとめ，今後の課題について述べる．

連絡先: 小俣仁美，東京工業大学情報理工学院情報工学系，〒 152-8552　東京都目黒区大岡山 2-12-1，komata@il.c.titech.ac.jp

2. ソフトウェアテスト

2.1 ソフトウェアテストの手法

ソフトウェアに誤りが含まれているかどうか確認するため，開発者は作成したプログラムに対しテストを行う．テストを行うことにより，ソフトウェアが正しく動作し安全なものであることを示すことができる．ソフトウェアテストの手法の代表的な種類として，ホワイトボックステストとブラックボックステストが知られている．ホワイトボックステストはソフトウェア内部の論理やプログラム構造を調べるものである．一方，ブラックボックステストはソフトウェア内部に関する情報なしにテストを行う手法である．

ブラックボックステストの手法として，同値分割，限界値分析，オールペア法などが知られている．同値分割とは，入力データについて有効な値と無効な値に分ける手法である．これにより，テストケースの数を減らし，ソフトウェアのエラーを識別するようなテストケースを得ることができる [Khan 12][Jovanović 06]．限界値分析とは，入力データに定められた変域の限界値に着目することにより，入力値に関して発生するエラーを発見するための手法である [Neate 06]．オールペア法は入力するパラメータの全ての組み合わせを網羅できるテストケースを用いるものである．そのようなテストケースを用いることにより，入力可能なデータ全てに対しソフトウェアテストを行うことが可能になる．

2.2 ソフトウェアテストケース生成

ソフトウェアテストを行うことによりソフトウェアに含まれる誤りを検出できる一方で，テストを行うことは開発者にとって負担の大きいものであり，ソフトウェア開発のコストの約 50%を占める [Myers 06][Abdel-Hamid 88]．そのため，ソフトウェアテストの自動化は開発コストを大きく削減することができる．

テストケースの生成を自動で行う手法の種類として，パスワイズテストデータ生成，データ仕様生成，ランダムテストデータ生成がある [Korel 90][Moadab 16]．

パスワイズテストデータ生成は，プログラムの制御フローから得られる制約を充足するようにテストケースを生成するものである．データ仕様生成はプログラムの機能からテストケー

スを生成するものであり，そのため生成にはプログラムの形式的な仕様を必要とする．ランダムテストデータ生成はランダムアルゴリズムでテストケースを生成する手法である．

3. 解集合プログラミング

3.1 概要

論理プログラム Π 中のアトムからなるある集合 M について，$\neg B(B \in M)$ を条件部に持つルール及び残りのルールの条件部に含まれる全ての負リテラルを Π から削除して得られるプログラムを Π_M とする．この時 Π_M はユニークで極小のエルブランモデルである．このモデルが M と一致する時，M は Π の解集合といい [Gelfond 88]，解集合を生成するプログラムを解集合ソルバーという．解集合プログラミングは解集合ソルバーのフロントエンドである．

解集合プログラミングは宣言型指向のプログラミングであり，NP 困難のような難しい探索問題に用いられる [Lifschitz 08]．また，解集合プログラミングは制約プログラミングの一種である．問題を論理プログラムとして記述したものを，問題の解が満たすべき制約式の集合とする．これに解集合ソルバーを適用することにより，制約式を満たすような解集合を探索する．得られた解集合が元の問題の解となる [Sakama 10]．

3.2 解集合プログラミングの記法

解集合プログラミングにおいて，ルールは次のように記述する．

$$A \leftarrow A_1, \ldots, A_m, notA_{m+1}, \ldots, notA_n \qquad (1)$$

A 及び A_i はアトムである．←の左側を帰結部，右側を条件部といい，条件部に記された条件 $A_1 \wedge \ldots \wedge A_m \wedge \neg A_{m+1} \wedge \ldots \wedge \neg A_n$ が真であるとき，帰結部のアトム A は真となる．

; はプーリングを行う記号である．$(\ldots, X; Y, \ldots)$ という引数のリストは $(\ldots, X)$, $(Y, \ldots)$ の省略形である．プーリングが帰結部で行われる場合は論理積として，条件部で行われる場合は論理和として引数リストが展開される．

4. 提案手法

4.1 概要

対象ソフトウェアのブラックボックステストに必要なテストケースを，入力データの仕様から自動で生成する手法を提案する．データ仕様からソフトウェアのテストケースを生成する場合，ソフトウェアに入力するデータの仕様を用意する必要がある．本手法では入力データの仕様を解集合プログラミングのルールとして記述することにより，解集合ソルバーを適用し，ルールに対する解集合としてテストケースを求める．解集合プログラミングを記述に用いることで入力データのパラメータの記述が容易になり，それを用いて各パラメータの入力値の組み合わせであるテストケースの集合を求めることができる．

まず，パラメータそれぞれについて，ブラックボックステストの同値分割や限界値分析の観点からテストすべきとされる値を挙げる．その後，それらの値を組み合わせることで，有効となるようなパラメータの組み合わせの入力データと無効となるようなパラメータの組み合わせの入力データを生成し，これらを合わせた集合を対象ソフトウェアのテストケースとする．

4.2 テストすべき値の列挙

各パラメータに与えることのできる値の中から．各パラメータに対してテストに用いるべき値を解集合プログラミングで列挙する．同値分割して得られる種類ごとの代表の値の他に，限界値の周辺をテストする値を用意する．

アルゴリズム 1 は，テストに用いるべき値をパラメータ毎に列挙するものである．同値分割に伴う各種類の値の代表値 e，限界値分析に伴う限界値 b を使用する．限界値周辺に対するテストを行うため限界値に与える微少変化量を delta/1 で宣言する．パラメータ p について入力データが有効となるような値の変域を，range/3 及び変域を定める不等号 $sign$ を用いて range(p, $sign$, b) と表す．限界値のチェックのため，パラメータ p の限界値に関する条件 range/3 をもとに，限界値に微少量の変化 Δ を与えて限界値周辺に関するテストのための値を作成する．

4.3 テストすべき値をもとにしたテストケース生成

テストに用いるべき値を組み合わせてテストケースを生成する．有効となるような入力データの条件から，有効となるパラメータの組み合わせのテストケースと無効となるパラメータの組み合わせのテストケースを生成する．この時，テスト実行時に期待される対象プログラムの出力結果も合わせて生成する．

アルゴリズム 2 は宣言した入力データの条件をもとにパラメータごとのテストに用いるべき値を組み合わせてテストケースを求めるものである．$\mathrm{condition}_1/j$, …, $\mathrm{condition}_k/j$ は，入力データに含まれている j 個のパラメータの値によって定義される k 個の述語であり，$\mathrm{condition}_1/j \wedge \ldots \wedge \mathrm{condition}_k/j$ が真となるような j 個の値の組み合わせを入力した時に開発者が期待するソフトウェアの出力値を $result$ とする．

5. 評価実験

提案手法がテストケース生成手法として有用であるかどうか，正しく機能するかどうかを適用事例を用いて確認した．以下の 2 点の観点から評価する．

評価 1
提案手法によって，対象ソフトウェアに必要なテストケースを生成することができたか

評価 2
提案手法によって生成されたテストケースの中に役割の重複するものがないか

適用事例および正解セットとして，Myers による三角形テストとその要件 [Myers 06] を用いる．解集合プログラミングとして clingo[cli] を使用する．

三角形テストで挙げられているプログラムに提案手法を適用し，テストケースを生成する．そして，正解セットの要件と提案手法により得られたテストケース集合を比較する．

評価 1 のために，Myers による三角形テストで必要とされる 14 個のテストケースの要件のうち，満たすことのできた数を確認する．また，評価 2 のために，条件ごとに合致するテストケースの内容を確認する．

5.1 適用結果

表 1 に，実験で利用したテストケースの要件と適用結果を示す．

組み合わせに関するテストケースを網羅することはできているが，小数に関するテストケースやパラメータの個数に関するテストケースのような各パラメータについてのテストケースの生成をすることができていない．また，提案手法ではテストに用いるべき値からテストケースを生成することはできるが，

Algorithm 1 テストに用いる値の列挙

Input: p: 対象パラメータ; e_i: パラメータ p に与えることのできる値の代表値; b: 限界値; Δ: 限界値テストに用いる微少量値
Output: value(p, v): パラメータ p についてテストで用いる値 v;
value(p, e_1; p, e_2; ...; p, e_i).
value(p, b; p, (b + Δ)) ← range(p, >, b), delta(Δ).
value(p, b; p, (b - Δ)) ← range(p, <, b), delta(Δ).

Algorithm 2 テストケースの生成

Input: value(p_j, v_j): 入力する対象のパラメータ p_j とテストすべき値 v_j; condition$_k$/i: 出力値が $result$ となるような入力データの組み合わせ (v_1, v_2, ..., v_j) に関する条件; $result$: 入力 (v_1, v_2, ..., v_j) によって期待されるプログラムの出力
Output: object(v_1, v_2, ..., v_j, $result$): テストケースの入力の組み合わせ (v_1, v_2, ..., v_j) と期待されるプログラムの出力 $result$
object(v_1, v_2, ..., v_j, $result$) ←
condition$_1$(v_1, v_2, ..., v_j), condition$_2$(v_1, v_2, ..., v_j), ..., condition$_k$(v_1, v_2, ..., v_j), value(p_1, v_1), value(p_2, v_2), ..., value(p_j, v_j).

各テストケースの役割の重複を検知して重複したものを削除することができないため，入力の候補となるパラメータの値が増加すると組み合わせにより生成されるテストケースも増えてしまう．

5.2 分析

評価 1
提案手法により必要なテストケースの多くを生成できた．一方で，解集合プログラミングは小数を表現することができないため，限界値分析で用いる微少量に小数を割り当てることができず，小数の入力を含むテストケースを生成することができなかった．また，入力データ中の値の個数の変更に対応できていないため，プログラムの引数に関するテストケースも用意することができなかった．

評価 2
提案手法では他のテストケースと役割が重複している不要なテストケースを減らすことができないため，テストケースが増大する恐れがある．本評価で扱った事例においては，三角形の種類の判別に関してエラーが含まれていないかどうかを調べる目的で入力する数値を複数必要とするため，それを元に生成できる入力値の組み合わせが増えることによりテストケースの数の増加につながっている．

5.3 結論

評価 1
テストに用いるべき値から組み合わせを作成し，テストケースを生成できた．しかし，解集合プログラミングで表現できない種類の数値があり，必要なテストケースを全て生成することはできなかった．

評価 2
テストにおける役割の重複を確認しないままテストケースを生成しており，不要なテストの実行につながる恐れがある．

6. 関連研究

テストケースの生成を支援する手法は多く存在する．例えば，動的最適化ベースの探索により限界値分析や実行時例外のテストを行う手法 [Tracey 98]，グラフを用いてホワイトボックステストを行う手法 [Wijayasiriwardhane 11] が提案されている．

SAT ソルバー [Yan 06] や制約解集合プログラミングを用いて組み合わせテストケースを生成する手法 [兼行 15] も存在するが，同値分割や限界値分析についてテストケースを生成するものではない．

7. おわりに

本稿では，プログラムで扱うオブジェクトの定義からブラックボックステストを自動的に生成する手法を提案した．提案手法では，同値分割や限界値分析について考慮すべきオブジェクトを解集合プログラミングで表現し，ブラックボックステストを行うために必要なテストケースの集合を求める．提案手法を Myers の三角形テストに適用し，プログラムの誤りの発見のために必要な入力を含むテストケースの一部を生成できることを確認した．

今後の課題は，限界値分析に必要な微少量の表現とテストケース削減手法の考察である．限界値分析や入力値の種類に関するテストのために，小数を扱うことのできる表現手法が必要である．解集合プログラミングの拡張として，解に小数を含むことのできる制約充足問題のソルバーが存在する [Balduccini 09a][Balduccini 09b] が，宣言に小数を用いることはできない．また，同じ役割のものがあるために不要となるテストケースを削減するため，テストの目的が重複しているテストケースを検出するアルゴリズムが必要である．

参考文献

[Abdel-Hamid 88] Abdel-Hamid, T. K.: The Economics of Software Quality Assurance: A Simulation-Based Case Study, *MIS Quarterly*, Vol. 12, No. 3, pp. 395–411 (1988)

[Balduccini 09a] Balduccini, M.: *CR-Prolog as a Specification Language for Constraint Satisfaction Problems*, pp. 402–408, Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[Balduccini 09b] Balduccini, M.: Representing Constraint Satisfaction Problems in Answer Set Programming, in *In: ICLP09 Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP09).(2009)* (2009)

[cli] clingo and gringo`https://potassco.org/clingo/`

表 1: テストケースに課される 14 個の要件と適用結果

番号	内容	生成の結果
1	不等辺三角形を表すテストケース	✓
2	正三角形を表すテストケース	✓
3	二等辺三角形を表すテストケース	✓
4	2 つの等辺を含む 3 種類の組合せ全てを試す二等辺三角形のテストケース	✓
5	1 つの辺がゼロをしめすテストケース	✓
6	1 つの辺が負をしめすテストケース	✓
7	ゼロより大きい 3 つの整数をもち，そのうち 2 つの和がそれ以外の 1 つと等しいテストケース	✓
8	7 項において，1 辺の長さが他の 2 辺の長さの和に等しいことを少なくとも 3 種類の順列のすべてに対してためすことのできるテストケース	✓
9	ゼロより大きな 3 つの整数のうち，2 つの数の和がそれ以外の 1 つの数より小さくなるテストケース	✓
10	9 項において，少なくとも 3 種類の順列すべてを考慮することのできるテストケース	✓
11	すべての辺が 0 であるテストケース	✓
12	整数でない値をもつテストケース	
13	数値の個数がまちがっていることをためすテストケース	
14	それぞれのテストケースについて入力の値に対して予想される出力	✓

[Gelfond 88] Gelfond, M. and Lifschitz, V.: The Stable Model Semantics For Logic Programming, pp. 1070–1080, MIT Press (1988)

[Jovanović 06] Jovanović, I.: Software testing methods and techniques, *The IPSI BgD Transactions on Internet Research*, Vol. 30, (2006)

[Khan 12] Khan, M. E., Khan, F., et al.: A comparative study of white box, black box and grey box testing techniques, *International Journal of Advanced Computer Science and Applications (IJACSA)*, Vol. 3, No. 6 (2012)

[Korel 90] Korel, B.: Automated software test data generation, *IEEE Transactions on Software Engineering*, Vol. 16, No. 8, pp. 870–879 (1990)

[Lifschitz 08] Lifschitz, V.: What is Answer Set Programming?, in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pp. 1594–1597, AAAI Press (2008)

[Moadab 16] Moadab, S. and Rashidi, H.: Automatic path-oriented test data generation by boundary hypercuboids, *Journal of King Saud University - Computer and Information Sciences*, Vol. 28, No. 1, pp. 82 – 97 (2016)

[Myers 06] Myers, J., Thomas, M., Badgett, T., Sandler, C., Nagao, M., and Matsuo, M.: ソフトウェア・テストの技法 第 2 版, 近代科学社 (2006)

[Neate 06] Neate, B.: Boundary value analysis, *University of Wales Swansea* (2006)

[Sakama 10] Sakama, C. and Inoue, K.: Answer Set Programming(¡Special Issue¿Current Trends in Logic-Based Inference Techniques), *Journal of Japanese Society for Artificial Intelligence*, Vol. 25, No. 3, pp. 368–378 (2010)

[Tracey 98] Tracey, N., Clark, J., Mander, K., and McDermid, J.: An automated framework for structural test-data generation, in *Proceedings 13th IEEE International Conference on Automated Software Engineering (Cat. No.98EX239)*, pp. 285–288 (1998)

[Wijayasiriwardhane 11] Wijayasiriwardhane, T. K., Wijayarathna, P. G., and Karunarathna, D. D.: An automated tool to generate test cases for performing basis path testing, in *2011 International Conference on Advances in ICT for Emerging Regions (ICTer)*, pp. 95–101 (2011)

[Yan 06] Yan, J. and Zhang, J.: Backtracking Algorithms and Search Heuristics to Generate Test Suites for Combinatorial Testing, in *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, Vol. 1, pp. 385–394 (2006)

[兼行 15] 兼行 大将, 番原 睦則, 宋 剛秀, 田村 直之, 井上 克巳：組合せテストケース生成問題に対する制約解集合プログラミングの適用, 2015 年度人工知能学会全国大会論文集 (2015)