2N-06

テストケース生成プロセスの自動アニメーション支援ツールの開発

須貝　健太　　劉　少英

法政大学情報科学部

Abstract

Testing-Based Formal Verification (TBFV) is a recently developed technique for testing programs against their specification. It is characterized by converting the testing of a program into the testing of a set of theorems each of which states what conclusion can be derived from what premises. The most important thing in TBFV is how to effectively generate test cases from a conjunction of premises. To address this problem, the process of generating a test case needs to be comprehensible to allow the tester to understand the effect of the currently proposed test case. In this thesis, we propose an animation approach to supporting the visualization of the test case generation process. We describe the principle of the animation and discuss how the supporting tool is built. We also show how the tool is tested to ensure its performance and compare our work with existing related work. Finally, we give our conclusion on our work and present some topics for future research.

1. まえがき

仕様からプログラムの正しさを反映する定理を導出し，その有効性をテストによって検証する Testing-Based Formal Verification(以下 TBFV)が提唱されている[1]. TBFV は事前条件，事後条件に基づいた検証をするため，ユーザーからの要求に沿った効率のよいテストが行えたり，テストデータの数を抑えたりすることができる．また，テストを用いた検証なので，自動化ができる．TBFV はどのようにして効率の良いテストデータが生成されるかが重要である．SOFL 形式仕様[2]に基づくアニメーションを行うツールが開発されているが，SOFL そのものの理解を深めるためのものであって，テストデータの生成はできない[3]．そこで本研究では，TBFV によってプログラムのテストデータ生成プロセスをアニメーションするためのツールを研究開発した．このツールでは，その生成過程を可視化することで，ユーザーが検証においてどのようなテストデータが有効かを調べることができる．

Development of an Automatic Animation Supporting Tool for Test Case Generation Process

Kenta Sugai, Shaoying Liu, Faculty of Computer and Information Science, Hosei University

2. TBFV

TBFV では SOFL(Structured Object-Oriented Formal Language)のように 1 つのプロセスに対して，事前条件と事後条件が示されている仕様を対象にする．仕様の事前条件から前提 H を，事後条件から結論論理式 C を導出し，H ⊢ Cを検証する．前提 H を満たすテストデータを用いて C がすべて真になれば，仕様に基づくプログラムからバグが発見されなかったといえる．逆に，前提 H を満たすテストデータを用いて C が偽になった場合，仕様に基づいて実装されたプログラムがバグを含んでいるといえる．次に Liu の論文[1]に紹介された TBFV の概念を紹介する．

2.1. 前提 H

選言標準形の入力に関する制約を前提 H とする．n を 1 以上の整数とすると，前提 H は次のように表せる．

$$H = P_1 \vee P_2 \vee \cdots \vee P_n$$

次に，テストデータを生成する．データのセット$T_i(i = 1 \ldots n)$に含まれるデータが前提 H を満たすかを調べる．前提 H を満たすデータをテストデータとする．

2.2. 結論論理式 C

事後条件から結論論理式 C を導出する．$y_1 \ldots y_m$をプログラム$\mathrm{Af}(x_1 \ldots x_n)$に対する返値とし，$\mathrm{Q}(y_1 \ldots y_m)$を事後条件とすると，C は次のように表せる．

$$\exists_{y_1 \in T_1, y_2 \in T_2, \ldots, y_m \in T_m} \cdot Af(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_m) \wedge Q(y_1, y_2, \ldots, y_m)$$

この式は，Af から条件 Q を満たすような$y_1 \ldots y_m$が返され，それらは仕様に示された型T_mに属するということを意味する．この式には，Af が関数のような形ではあるが，必ず数式で表現できるわけではない．このような関数を含む定理H ⊢ C は，論文[1]で初めて提案された．TBFV では，プログラムを呼び出し，その結果が事後条件を満たしているかを調べることで，プログラムにバグが含まれているかを判断する

3. テストデータ生成アニメーション

テストデータ生成アニメーションとは，テストデータの生成プロセスを可視化するものである．ここでは，テストデータ生成アニメーションの設計について記述する．SOFL 形式仕様を解析した結果を用いてアニメーションの描画をする．アニメーションは 1 つの論理式を状態として扱い，状態遷移図のような表現をする．入力された値が条件を満たすかどうかを調べ，満たせば次の条件に遷移する．

3.1.アニメーション画面設計

まず，条件とは関係ない状態を初期状態として，画面の左端に設置される．この状態の位置には，前提 H に関する状態が遷移する様子を表すための赤い円が*P*の数だけ設置される．前提 H に”or”が含まれる場合，初期状態から上下に分岐した後，*P*に含まれる原子論理式の数だけ横に状態が並べられる．前提 H の条件すべてについて状態が並べられた後，初期状態と同じ高さの位置に 1 つの状態を設置する．この状態の位置には，結論論理式 C に関する状態が遷移する様子を表すための青い円が設置される．この状態の右に結論論理式 C についての状態を前提 H と同じように並べる．例として H = x1 > 0 and x2 > 0 or x1 < −1，C = y > x1 + x2 というように前提 H と結論論理式 C が定義された場合について説明する．この場合，初期状態から上下に 2 つの分岐が起こり，上の分岐先には”x1 > 0”と”x2 > 0”を表す状態が横に並ぶ．下の分岐先には”x1 < −1”を表す状態が設置される．“x2 > 0”の右側で，初期状態と同じ高さの位置に状態が 1 つ設置され，その右側には”y > x1 + x2”を表す状態が設置される．アニメーション画面の例を次の図 1 に示す．

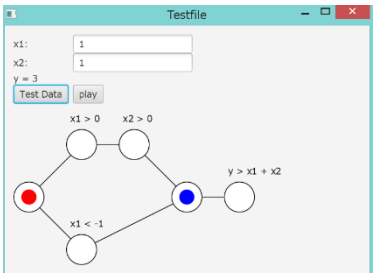

図 1　アニメーション画面の例

3.2.アニメーション動作設計

アニメーション動作の設計について記述する．まず，入力された値を取得する．SOFL 形式仕様を解析した際に取得した入力変数の情報から，入力フォームの数と，その変換先の型を決める．GUI の入力フォームでは，入力した値が文字列として扱われるので，論理式が正しく評価できるように変換する．次に，入力値に対するそれぞれの論理式の真偽を確かめる．前提 H から取得した論理式に入力値を代入する．前提 H に含まれる*P*が 1 つでも真となれば，テストデータの生成が成功したといえる．テストデータ生成に成功したとき，赤い円が初期状態の位置から，真となる論理式の状態を通って青い円の位置まで移動する．最後に，生成されたテストデータを用いて結論論理式 C の評価を行う．前提 H と同じように，論理式が真となれば青い円が次の状態へ遷移する．この評価では，SOFL 形式仕様に基づいて実装されたプログラムの関数を呼び出し，実行させている．アニメーションの流れを図 2 に示す．

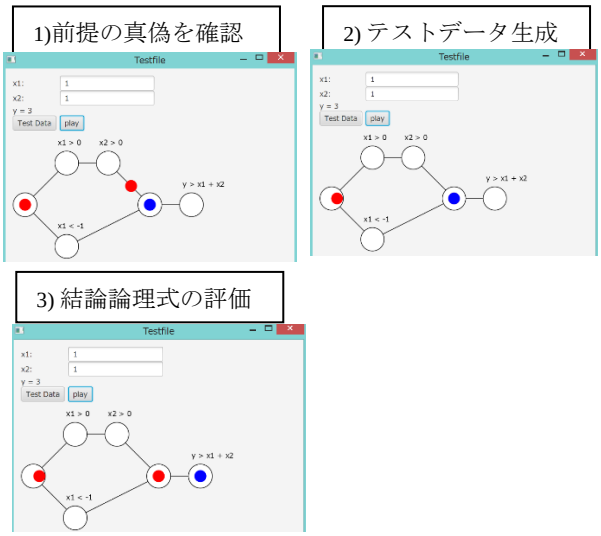

図 2　アニメーションの流れ

1)は入力された値について論理式の真偽を調べ，赤い円を遷移させている．図 2 の例では，”x1 > 0”と”x2 > 0”は真になるが，”x1 < −1”は偽になるため遷移できない．2)は前提 H についての遷移が終了し，テストデータの生成ができたことを表している．3)はテストデータを用いた結論論理式の評価を行っている．

4. むすびと今後の課題

論理式を用いた事前条件と事後条件に基づいてテストデータの生成が行えるアニメーションツールを開発した．状態遷移図のアニメーションを用いることで検証内容を具体的に伝えることができた．このツールの発展としては，テストの自動化が望まれる．また，現在のツールの機能では，データストアなどの外部変数へのアクセスを含むプロセスについて検証できない．この問題を解決するには，SOFL 形式仕様を読み出す手順において，データストアなどの記述に対応し，それらに適した変数やクラス定義をテストプログラムに実装できるようにしなければならない．そのためには，読み込んだ文字列の SOFL 形式仕様における意味を厳密に理解するようなコンパイラに近いものを実装する必要がある．また，テストデータの自動生成ツール[4]の機能と組み合わせればシステム全体の自動化ができる．これらを今後の課題とする．

文　献

[1] Shaoying Liu, “Testing-Based Formal Verification for Algorithmic Function Theorems and Its Application to Software Verification and Validation”, 2016 International Symposium on System and Software Reliability, 2016.

[2] Shaoying Liu, Formal Engineering for Industrial Software Development Using the SOFL Method, Springer- Verlag, 2004.

[3] 中村悠士, “SOFL 形式仕様アニメーションの支援ツールの開発”, 法政大学卒業論文, 2016 年

[4] 池田逸人, “形式仕様に基づくテストケース自動生成とテスト結果の自動評価”, 法政大学修士論文, 2017 年