プログラム変更に対する正当性検証技法の適用

2K−2

丸山 勝久　小野 康一　門倉 敏夫　深澤 良彰
早稲田大学 理工学部

1　はじめに

既に開発されて稼働中のソフトウェアが、保守要求等によって変更される場合を考える。これをシステム変更と呼ぶことにする。いま、システム変更において検証を行うとき、以下のことを仮定する。

- 変更前のプログラムは、検証によって変更前の仕様に対する正当性が保証されている。また、この検証の結果(履歴)は残っている。
- 保守要求によって変更された仕様は、仕様定義者の意図を正確に表現している。また、仕様は誤りを含まない。

保守要求によって、仕様とプログラムが個々に変更されると、変更後においてはこれらの間で正当性が満たされる保証はない。したがって、変更後の仕様とプログラムに対して再度検証を行う必要がある。本稿では、このようなシステム変更に対する検証技法の適用について述べる。

2　目的

システム変更に対する検証を考えたとき、以下に示すことがいえる。

「検証対象の一部だけが変更を受けるため、検証対象の大部分は変更後の検証に影響を与えない。」

このことから、変更前の検証結果を変更後の仕様とプログラムの検証に利用することが考えられる。本検証系では、システム変更前の検証結果と変更後の検証対象を比較することで、仕様やプログラムの変更が検証対象のどの部分に反映するのかを特定することができる。このことから、以下に示す利点が得られる。

- プログラム中の誤りを含む箇所を特定できることがある。
- これから変更すべき機能だけを仕様から抽出することができる。よって、不必要な変更を抑えることになる。

このように、本検証系は、正当性の検証技法を用いることで、プログラム変更に対する支援を行い、その労力の削減を図ることを目的としている。

3　本検証系の概要

本検証系の構成を図1に示す。

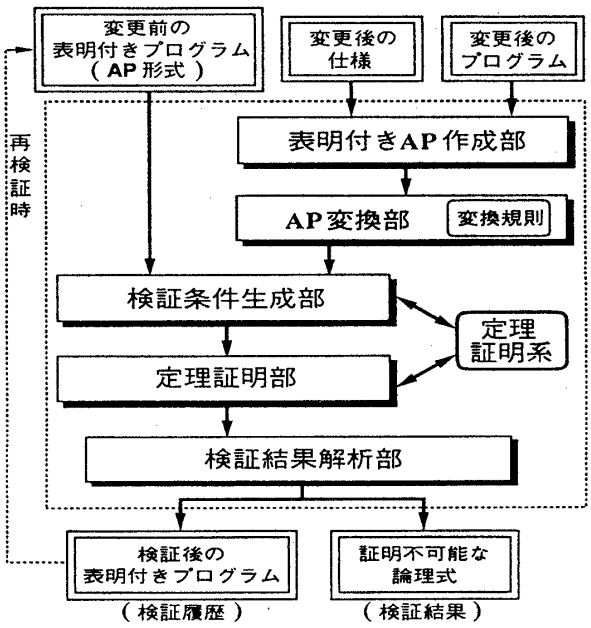

図1　本検証系の構成

3.1　入出力

検証対象として、要求仕様とプログラムを用意する。仕様記述言語については、定理証明手法を用いることを考慮して、一階述語論理に基づくものを考える。検証の際に、この仕様から入力表明、出力表明を作成する。また、プログラムについては、一般的な手続き型言語を用いる。ただし、ここで扱うプログラムは、再帰構造をもたないものとする。プログラムにループが存在する場合は、プログラマが不変表明をプログラム中に記述する。さらに、本検証系はシステム変更前の検証結果を変更後の仕様やプログラムの検証に利用するため、変更前の表明付きプログラムも入力する。

プログラム変更を支援するために、正当性の真偽だけでなく、証明が成功しなかった論理式についても、仕様やプログラムと対応させて出力する。この出力のうち検証履歴は次に行なう検証の入力となり、証明が成功した部分を次の検証で用いる。

3.2　拡張AP

従来の検証系と異なり、本検証系は検証条件を生成する前に表明付きプログラムの等価変換を行う。この等価変換を行う前に、プログラムからAP(Abstract Program)を作成する。APはMannaが提唱したもので、プログラムと等価な論理式付き有向グラフである。

本検証系では、AP変換の際にプログラム中の経路で並列に実行可能なものを表すために従来のAPを拡張する。

An application of the verification method for program modification
Katsuhisa MARUYAMA, Kouichi ONO,
Toshio KADOKURA, Yoshiaki FUKAZAWA
School of Science and Engineering, WASEDA University

拡張APでは節点に対して、分岐が2種類存在する。一つは従来の分岐で、複数の経路のうちどれか一つを選択するものであり、それらの経路はOR関係にあるといえる。もう一つの分岐は並列関係を表すもので、その節点に接続されている経路が並列に実行可能なことを示す。これらの経路はすべて実行しなければならないので、AND関係にあるといえる。これを表すために矢印の間に弧を付けることにする。これによって、APはOR有向グラフからAND-OR有向グラフに拡張されたことになる。

3.3 処理内容

図1の各部の処理内容について、以下に簡単に示す。

(1) 表明付きAP作成部

Floyd-Hoareの帰納表明法に基づき、逆向代入法を用いることによって、仕様と不変表明を含むプログラムから表明付きAPを作成する。

(2) AP変換部

APが一意になるように、APの変換を行う。AP変換としては、次の2種類を用意する。

(a) 表明を利用する論理的等価変換

(b) 同時に実行可能な部分を並列化する変換

(a)により、節点間の接続関係をより簡潔にすることができる。また、これによって、並列化に対する前準備も同時に行うことになる。

(b)の並列化は、プログラム内の変数の干渉状態を考慮して行われる。ここで、同時に実行可能な経路は、拡張APのAND関係で表現される。この並列化によって、プログラムが分割されるため、検証条件生成時に変更前のAPと変更後のAPで一致する箇所が増えることになる。

これら2つの変換はあらかじめ用意した規則をAPに適用することで達成される。具体的には、APと変換規則でマッチングを行い、マッチングが成功した際に、変換規則に従ってその部分を書き換える。以上の操作を規則が適用できなくなるまで繰り返す。

(3) 検証条件生成部

変更前のAPと変更後のAPを比較し、定理証明系を用いて変更後のAPを簡約する。この操作を以下に示す。また、APの簡約例を図2に示す。

(i) 変更前のAPの節点の接続関係と同型な部分を変更後のAPからグラフ・マッチングによって抽出する。このとき、矢印に割り当てられている論理式(テスト述語と代入式)も一致している必要がある。

(ii) マッチするサブグラフが見つかったとき、そのグラフに対して外部から入力している矢印の元の入力節点と外部へ出力している矢印の先の出力節点を見つける。

(iii) 対応する入力節点について、変更後の論理式から変更前の論理式が導出可能であるかどうかを調べる。

(iv) 導出可能なとき、そのサブグラフをAPから削除する。また、対応する出力節点についても、変更前の論理式から導出可能なものを変更後の論理式から削除する。

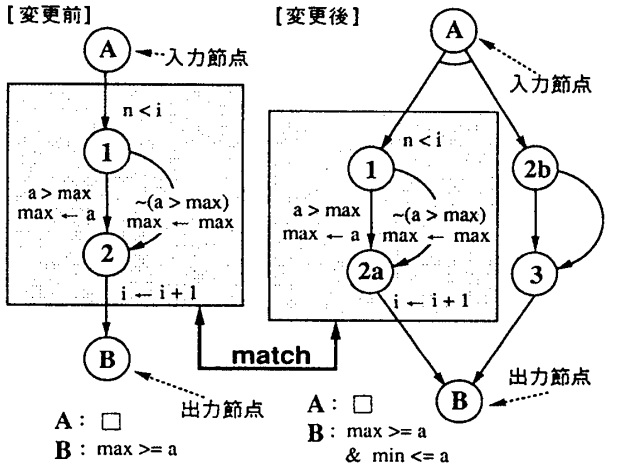

図2 AP簡約の例

以上の操作を、同型部分がなくなるまで繰り返す。簡約後のAPに対する検証条件生成法は従来の方法と同じである。

(4) 定理証明部

生成された検証条件に対して、定理証明系を用いて実際に証明を行う。

(5) 検証結果解析部

検証履歴として、証明可能な論理式をAPに対応づける。また、証明不可能な論理式に対しても、検証結果として、仕様やプログラムへの対応づけを行う。

現在、定理証明系については、既存のものとしてGoodらの検証システム[1]で用いられている定理証明器を使用している。

(1)~(5)の処理によって、証明不可能な論理式が空、つまり検証結果がすべて真となるとき、仕様とプログラム間の部分正当性が成立することになる。また、証明に失敗した場合、証明不可能な論理式(検証結果)はこれから変更すべき機能を表し、検証後の表明付きプログラム(検証履歴)はいままで変更してきた箇所を示している。

本検証系では、この2つの出力によって、誤り箇所や変更が不足している箇所をプログラマが推測するのを支援する。変更後の仕様に対してプログラムが正当性を満たすまで、プログラム修正と検証を繰り返す。本検証技法によって、プログラムを変更してから検証を行うのではなく、検証してからプログラム変更を行うという流れになる。

4 おわりに

現在、この検証系の有用性を調べるために、実際のシステムを作成中である。今後の課題として、本検証系を拡張し、プログラム変更の際の誤り箇所をさらに特定する手法を考えている。

参考文献

[1] Good,D.I., London,R.L. and Bledsoe,W.W., An Interactive Program Verification System, IEEE, Trans. on SE, vol.1, no.1, pp.59-67, March, 1975