構文木の比較に基づいたプログラム差分の表示方式

3H－4

会沢 実† 練 林† 飯田 元† 井上 克郎† 鳥居 宏次††

†大阪大学 ‡奈良先端科学技術大学院大学

1 まえがき

大規模なソフトウェアのソースプログラムは，開発，保守のそれぞれの過程を通じて頻繁に変更を加えられる．変更されたプログラムの履歴からバグの作り込まれた時間の特定などの解析を行なう際，バージョン間の差分を求めるツールは非常に有用なものである．開発者は，プログラムテキスト間の差分を基にどの部分に変更が加えられたかを知ることができる．

このようなプログラムの差分を求めるツールとしては，UNIX の diff[1] などが広く用いられている．diff では行単位のストリング比較を基にして差分を求めている [2]．しかし，C のプログラムのようにブロック構造を持つテキストの差分を求める場合，ネストの深さの変化や制御の流れの違いなども開発者にとっては重要な情報である．diff の差分の計算方法では，このようなプログラムの構造に関わる変化の有無をとらえることは困難である．

本研究では，構文木の比較を基にしたプログラムテキスト比較ツール (progdiff) の試作を行なった．progdiff では，ユーザがプログラムの階層構造の変化を容易に把握できるよう，表示方法についても工夫している．

progdiff の入力は，C 言語で記述されたプログラムテキストである．progdiff は主に次の 3 つの部分から構成される．(1) 入力された 2 つの C プログラムからそれぞれに対応する構文木を生成する．(2) 木の比較アルゴリズムを利用して，2 つの構文木の各頂点の最大の対応を求める．(3) 対応のとれていない頂点を両者の差分として，プログラムテキストの形で表示を行なう．

次章以降において，ツールを構成するそれぞれの部分について説明する．

2 構文木

progdiff が用いる構文木は，次のような特徴を持つ．

- 木に含まれる各頂点のラベルはプログラム中の各実行文 (代入文，if 文等の制御文)，変数宣言文，関数定義部，ラベルのそれぞれに 1 対 1 に対応づけられている．但し，木の根に当たる頂点にはラベルはない．
- 木に含まれる各内部頂点は，根を除くと，関数定義部，繰り返し文，条件分岐文のいずれかに対応している．
- 生成された木は，preorder で辿ることによって元のプログラムテキストに復元することが可能である．

ここでは，コンパイラが行なう構文解析のような，各文の詳細な解析に基づく構文木にはせず，実行文や宣言文を 1 つの頂点としている．このような，文や宣言毎の差分の方が，ユーザが直観的に変更を把握し易いと考えられる．また，余分な頂点の対応を求める必要がなく計算時間を短縮できる．

ファイル中に含まれるコメント文などは構文解析時に読み飛ばされ結果に影響は与えない．

3 アルゴリズム

プログラムの構文木間の差分を求めるために次のアルゴリズムを利用する．

SSPM(Strongly Structure Preserving Mapping) では木から木への自然な変換に基づく 2 つの木の間の距離を定義している [3, 4]．この距離の計算法を利用すると，木の頂点間の最大対応が求められる．筆者らは，SSPM に基づいた 2 つの木の間の最大対応を直接求めるアルゴリズムを提案した [5]．アルゴリズムの詳細については [5] において説明されているため，ここではその計算法については省略する．

このアルゴリズムを前章の構文木に適用して同じラベルを持つ頂点の組の最大集合を求める．求まった集合に含まれない頂点は削除または挿入されたものとみなす．

例えば，図 1 に示すような 2 つのプログラムの構文木 T, T' の場合，SSPM に基づく頂点の最大の対応は {(1,1'), (2,2'), (3,4'), (5,5'), (6,6')} となる．(4,3'), (7,7') はラベルが等しいがこれらを対応の集合に入れると SSPM の条件を満たさなくなる．従って差分としては，この場合 T から 4, 7 が削除され，T' へ 3', 7' が挿入されたと考える．

4 表示部

上記のアルゴリズムで得られた差分を分かり易く表示するために以下の検討を行なった．

- 表示の形式

  プログラム中の制御の流れを分かり易くするため，差分のみを単独で表示するのではなくプログラム全

A Display Method of Program Difference Based on Parsing Tree Comparison

Minoru AIZAWA†, Lin LIAN†, Hajimu IIDA†, Katsuro INOUE† and Koji TORII††

†Osaka University

‡Nara Institute of Science and Technology

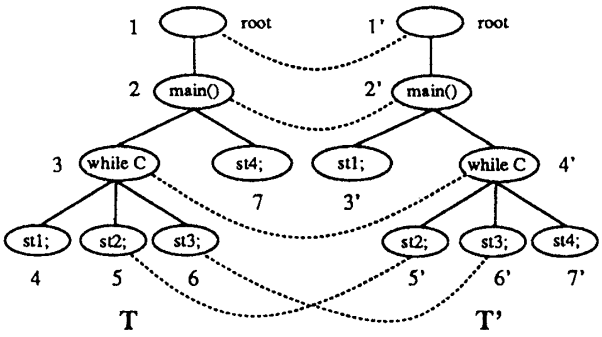

図 1: プログラムから得られる構文木の比較

体あるいは関数全体を表示する.

プログラムを画面上に表示する際, 構文木を基に直接木の形で表示する方法とテキストの形で表示する方法との大きく 2 つの分類が考えられる. ここではサイズの大きいプログラムに対処することも考え, 表示に要するスペースが比較的少なくて済むプログラムテキストの方を採用した. 比較した 2 つの構文木からプログラムテキストに復元し, 表示を行なった.

- テキストの配置

  比較した 2 つのプログラムは, progdiff を起動した同一ウィンドウ内に, 縦に 2 列に並行して表示することにした. 対応のとれている部分については, その部分の位置を横に揃える. これによって各関数や繰り返し文など対応のとれているブロックの位置が揃い, プログラムの階層構造の変化の有無をより把握し易くなる.

- 差分の表示

  変更部分 (削除または挿入された部分) は変更のない部分 (対応のとれている部分) と容易に区別できなければならない.

  本研究ではユーザが変更部分を把握し易いように, 余分な記号等を付加しない方法で差分の区別を試みた. そのため, カラーディスプレイ上での色の区別を利用した. 変更部分はテキストの色の違いによって変更のない部分と区別される.

- 関数の表示

  ユーザの選択によって, 変更部分を含む関数のみを表示し, 他の関数に関しては省略することのできる機能を付加した. 大規模なソフトウェアの開発においても, 修正作業を行なう場合は, その変更は部分的であることが多い. この機能により, 大規模なプログラムテキストを効率的に表示することが可能である.

  この場合, ファイル中における関数の相対的な位置が分かるよう, 変更のなかった関数の定義本体の各文のみを省略し, その関数名の宣言などの表示は残しておく.

5 むすび

前章で述べたような表示方法を基に progdiff を試作した.

ユーザが選択できるオプションとして,

- 2 つのプログラムのうちの片方のみを選んで表示.
- 変更のない関数を省略して表示.
- 白黒ディスプレイ上で色の区別の代わりに, 変更部分に下線を付して区別.

がある.

実際の progdiff の実行結果を図 2に示す.

progdiff により, ネストの深さの変化や制御の流れの違いなどプログラムに関するより詳しい情報を開発者に提供できる.

現在, 表示部を中心に評価方法を検討中である.

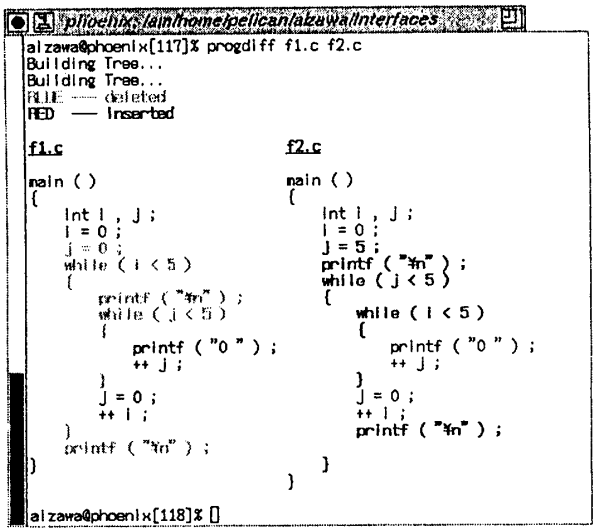

図 2: progdiff の出力例

参考文献

[1] B.W. Kernighan and R. Pike : "The UNIX Programming Environment", *Prentice Hall*, 1985.

[2] 角田 博保 : "ファイル間の相違検査法", 情報処理, Vol.24, No.4, pp.514–520 (April 1983).

[3] 田中 栄一, 田中 圭子 : "木の間の距離とその計算法", 電子通信学会論文誌 (D), Vol.J65-D, No.5, pp.511–518 (May 1982).

[4] 田中 栄一 : "強構造保存写像に基づく木の間の距離とその計算法", 電子通信学会論文誌 (D), Vol.J67-D, No.6, pp.722–723 (June 1984).

[5] 練 林, 井上 克郎, 鳥居 宏次 : "構文木間の対応に基づくプログラムテキスト比較ツールの試作", 電子情報通信学会技術研究報告, SS93–18, pp.33–99 (July 1993).