アルゴリズム認識のための プログラム共通部分構造の抽出法に関する一考察

6T−4

山田宏之　相原恒博
愛媛大学工学部情報工学科

1．はじめに

ソフトウェア開発支援システムが開発対象のソフトウェアを理解できると，より知的な支援を実現することができる[1]．そこで，本稿では，プログラム理解をめざし，対象プログラムの構造中に存在するアルゴリズムの構造を抽出する手法を考察する．

2．アルゴリズム認識

アルゴリズムとは，ある問題を有限時間内に解くための処理手順である．したがって，あるアルゴリズムを用いて作成されたプログラム内には類似した処理手順が含まれていると考えられる．

しかしながら，コーディングの多様性（ある機能を全く異なったプログラミングプリミティブで実現できること，プログラムに現れる識別子は同じ名前でも役割が異なったり，異なった名前でも同じ役割をもったりすること等）のため，単純にソースコードを比較しただけでは，そのアルゴリズムを認識することは容易でない．そこで，本研究では，ある一定の規則に基づいてプログラムをグラフ構造で表現することにより多様性に対処する．その結果，同一のアルゴリズムで実現されたプログラムのグラフ表現内には類似した構造が現れ易くなる．そこで，グラフ表現から共通部分構造（部分構造グラフ）を抽出し，それを抽象化してプログラム・スキーマを生成する．したがって，本研究のアルゴリズム認識はプログラムからプログラム・スキーマを認識することである．

3．プログラムの構造マッチング

3．1　対象プログラムの構造

本稿では対象プログラム記述言語として，Ｌｉｓｐ言語のサブセットを採用している．この言語は，リスト処理関数，条件分岐関数，算術演算関数，述語関数，配列用関数から構成される．そのうえ，プログラム（関数）は再帰的に定義されているものを対象とする．

再帰的に定義されたプログラムは，その引数の値により，再帰を停止し値を返す再帰停止部と問題をより簡単にして再帰呼出しをする再帰呼出し部にわけられる．そこで，引数の値に基づく分岐条件による場合分けに着目し，プログラムの内部表現を次に考察する．

3．2　プログラムのグラフ表現

Ｌｉｓｐ言語ではプログラムは複数の関数により定義される．そこで，プログラムの内部表現として，条件分岐による場合分け毎に，関数呼び出しをノードに，データの流れをリンクで表現するグラフ表現を採用する．リンクの向きがデータの流れを表現している．

本表現のノードとして，関数ノード，再帰呼び出しノード，接続を表す中間ノード，入力変数を表す入力ノードおよび出力ノードがある．ここで，例として平方根を２分探索法で求めるプログラムをとりあげ，その関数定義を図１に，そのプログラム・グラフを図２に示す．

```
(defun sqrt2 (n a b)
  (let ((c (floor (+ a b) 2)))
       (if (guess n c) c
           (if (< n (sqr c))
               (sqrt2 n a c)
               (sqrt2 n c b)))))
```

図１．平方根を求めるプログラムの定義

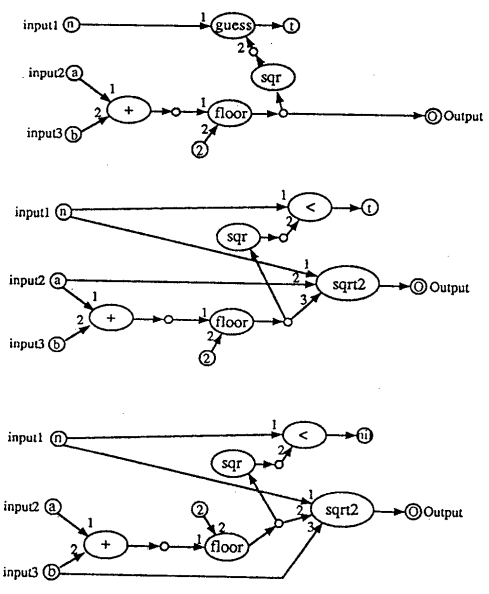

図２．関数sqrt2のプログラム・グラフ

A method of extracting common structure from programs implemented with the same algorithm for algorithm recognition

Hiroyuki YAMADA, Tsunehiro AIBARA
Faculty of Engineering, Ehime University

このとき，関数ノード，再帰呼び出しノードの入力についている添え字は呼び出し時に与えられる引数の順番を表している．また，図１中の①，②，③はそれぞれ図２中の(a),(b),(c)に対応する．

３．３　構造マッチングアルゴリズム

プログラム・グラフg_1，g_2内の共通する構造を次の手順で抽出する．

1）グラフg_1，g_2内のすべての関数ノード間の対応を調べる．このとき，引数の数，関数名，関数の型（再帰関数，算術演算関数，述語等）などの相違点をもとに照合評価規則によりペナルティを計算する．

2）ペナルティの小さい対応を一つ選び，それを初期対応（f_1，f_2）とする．また，全てのノードの対応の検査が終了しているならば，停止する．

3）それぞれの関数の入出力リンクをたどり，リンクの先のノード間の対応のペナルティを調べる．

4）ペナルティの低い対（f_{11}，f_{21}）を一つ選び，部分構造グラフ生成規則にしたがい，部分構造グラフを成長させる．成長した場合は成長した先の対からさらにグラフを成長させるために3)に戻り，3)，4)を成長しなくなるまで繰り返す．また，成長に失敗した場合または全ての対のペナルティが規定値より高い場合は，2)に戻り次の候補について同様の処理を行う．

部分構造グラフ生成規則の例を以下に示す．

1）f_{11}，f_{21}とも関数ノードの型および関数名が同じならば，再度ペナルティを計算し，その接続関係を保持し，部分構造グラフを成長させる．

2）f_{11}，f_{21}とも関数ノードの型が違う場合には，いづれかにダミーノードを入れ，その先のノードとダミーを入れなかったノードとのペナルティを調べる．

ペナルティが小さいときはダミーノードを採用し，部分構造グラフを成長させる．そうでない場合は，その対の照合に失敗する．

ここでは，マッチングの具体例として２分探索により，与えられた値の格納場所を求めるプログラムをとりあげ，その関数定義を図２に，プログラム・グラフの一部を図３に示す．また、この構造マッチングにより獲られた部分構造グラフの一部を図４に示す．

４．むすび

本稿では，プログラム理解をめざし，対象プログラムの構造中に存在するアルゴリズムの構造を抽出する手法を述べた．今後の課題としては，抽出された部分構造グラフからプログラム・スキーマの生成とそれを用いたアルゴリズム認識機構の開発がある．

```
(defun search2 (n a b)
  (let ((c (floor (+ a b) 2)))
       (if (= a b) nil
           (if (= (aref test c) n) c
               (if (< n (aref test c))
                   (search2 n a c)
                   (search2 n (1+ c) b))))))
```

図３．関数定義

［参考文献］

[1] 上野：”知的プログラミング環境－プログラム理解を中心に－”，情報処理，Vol.28，No.10，pp. 1280-1296，1987.

[2]電総研人工知能グループ他訳：”類推学習”，共立出版，pp.53-89，1988

[3] Winston, P.H. and Horn, B.K.P : "LISP", 3rd Edition, Addison Wesley, 1989.

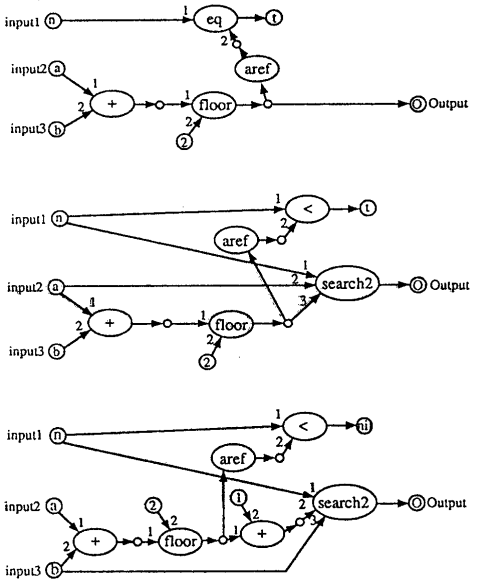

図４．例題のプログラム・グラフの一部

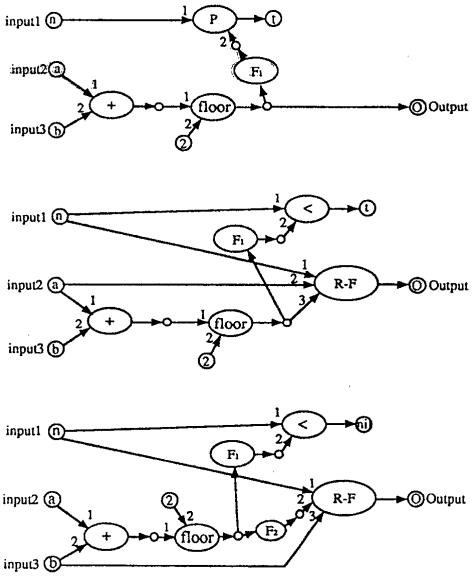

図５．部分構造グラフ