

Pascal対応Hichartの属性グラフ文法の拡張と

2L-8

オートフローチャータ

大井裕一[†], 安達由洋[†], 土田賢省[†], 夜久竹夫^{††}[†]東洋大学, ^{††}日本大学

1. はじめに

ISO Pascalに準拠した階層型プログラム図式言語Hichart^[1]を定義する属性グラフ文法^[2, 3]の拡張とオートフローチャータの高速化について報告する。

これまでの属性グラフ文法は、Hichart図を描画するための属性のみを定義していたが、これに、宣言された変数を格納する属性と、Pascalプログラムの行番号を情報として持つ属性を新たに加えることで属性グラフ文法を拡張する。

これにより、ソースプログラムとHichart図のセルとの間での相互参照が容易になるだけでなく、変数のチェックができるなど、より実用的で高度なプログラム開発環境を持つオートフローチャータが実現できる。また、属性の数を増やすと、Hichart内部表現を生成するための処理時間が長くなるので、属性評価器の実装法の再検討を行う。その結果、非常に高速なオートフローチャータが実現できることを示す。

2. 属性グラフ文法の変更点

本研究では、これまでに定義したHichart図表示のための属性に加えて、対応するソースプログラムの行番号を格納する属性と、変数をチェックするための属性を新たに定義する。また、図表示のための属性評価器をより高速にするためにこれまでの属性の一部を変更する。

以下にこれらの属性の一部を示す。

継承属性

top : 部分木内で最も上に配置されるセルのY座標
extern : 宣言された外部変数の集合
declared : 宣言された内部変数の集合
used : 宣言された内部、外部変数の集合

合成属性

y : 配置されるセルのY座標
bottom : 部分木内の最下辺のY座標
updated : すでに宣言された内部変数の集合と、新しく宣言される内部変数の集合との和集合
linenumber : 対応する行番号の集合

3. 変数のチェック機能

我々が実現した属性による変数のチェック機能は、文献[4]の手法を入れ子になった構造を持つプログラムにも対応できるように拡張したものである。

変数のチェックで用いる属性は次の4つである。

(a)extern, (b)declared, (c)updated, (d)used

(a)は外部変数のリスト、(b)はすでに宣言された内部変数のリスト、(c)は(b)の変数のリストと新しく宣言される内部変数を合成したリスト、そして(d)は外部変数と内部変数とを合成したリストで、プログラムで使用されている変数の宣言済み、二重宣言、未宣言等を判断するために使用される。

ここで変数チェックの流れを示す。

```

procedure 変数のチェック;
begin
  procedure 変数宣言 (declared, updated);
  begin
    Vs ← 内部で宣言される変数の集合;
    temp ← #;
    while Vs ≠ # do
      begin
        Vs ← Vs - {v}; {Vsの要素vを取り出す}
        if v ∈ declared ∪ temp then {二重宣言}
          エラーメッセージの出力
        else
          temp ← temp ∪ {v}
        end;
      end;
    updated ← declared ∪ temp;
  end;

  procedure 手続き宣言 (extern);
  begin
    変数宣言リスト (extern, updated);
    used ← declared ∪ updated;
    文リスト (used)
  end;

  procedure 文リスト (used);
  begin
    文 (used);
    if 次の文リストが存在する then
      文リスト (used)
    end;
  end;

  procedure 文 (used);
  begin
    while 文の中で他の変数が存在する do
      begin
        文の中に存在する変数を1つ取り出し、
        それをvとする;
        if v = 変数 then
          if v ∈ used then {未宣言}
            エラーメッセージの出力
          end;
        end;
      end;
    end;
  end;

  変数宣言 (#, updated);
  手続き宣言 (updated);
  文リスト (updated)
end.

```

図3.1 変数チェックを行う手続き

An Extension and Application of an Attribute Graph Grammar of Hichart for Pascal

Yuichi Oi[†], Yoshihiro Adachi[†], Kensei Tsuchida[†], Takeo Yaku^{††}

[†] Faculty of Engineering, Toyo University

^{††} College of Humanities and Sciences, Nihon University

次に例として、手続き宣言を1つ含むようなプログラムでの導出木と各ノードが持つ属性を示す。属性の上の数字はその属性が評価される順番を表す。

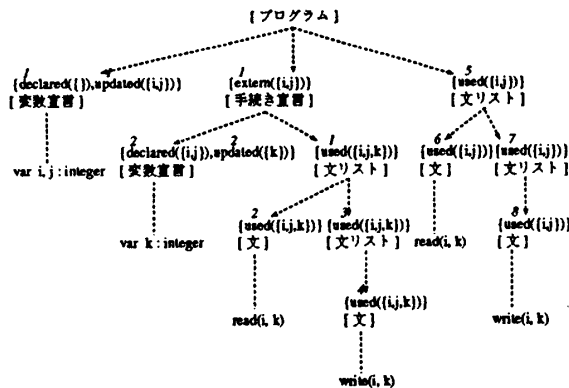


図3.2 導出木と属性の評価順序

4. 行番号の付加

これまでのHichartの内部表現にソースプログラムの行番号の集合を持つ属性 *lineno* を付加した。画面には、Hichart図に対応するソースプログラムも同時に出力し、セルをマウスでクリックすると、クリックされたセルとソースプログラムの対応する行がハイライトされる。この機能により、Hichart図とソースプログラムとの間の対応関係が容易に認識できるようになった。

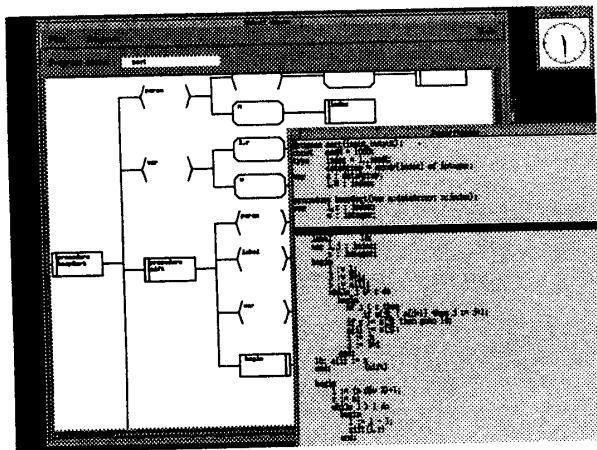


図4.1 Hichart図とソースプログラムとのリンク

5. 属性評価器の高速化

より高機能で実用的なオートフローチャータを実現するために、上記のように属性を新しく定義したが、これまでの属性評価器の手法では計算時間が長かった。その大きな理由はPrologのユニフィケ

ーションを用いて属性値を求める計算式をすべて生成し最後に一括してその値を求めていたことによる。そこで今回は評価式の引数の値が決まるとすぐに属性値を求めることで、長い式のコピーを無くし、評価時間を大幅に削減することができた。

図5.1に評価時間の比較をしたグラフを示す。横軸はHichart図のセルの数、縦軸は評価時間を表す。

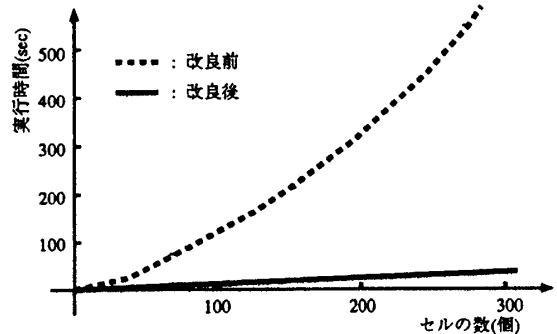


図5.1 解析時間の比較

6. おわりに

本研究ではPascal対応Hichartを定義する属性グラフ文法の拡張と、属性評価器の改良を行った。属性グラフ文法をHichart図の表示だけでなく、ソースプログラムとHichart図とのリンク付けや、変数チェックにも利用できるように拡張することでより実用的なオートフローチャータが実現できた。さらに、属性の評価法の改良により高速な属性評価器の生成が可能となった。

今後、C言語や、木構造図のデータ交換言語であるDXL^[5]に対応したオートフローチャータの実現を目指していきたい。

参考文献

- [1] YAKU, T., FUTATSUGI, K., ADACHI, A. and MORIYA, E.: HICHART—A Hierarchical Flowchart Description Language, Proc. IEEE COMPSAC 11, pp.157–163(1987)
- [2] NISHINO, T.: Attribute Graph Grammars with Applications to Hichart Editors, Adv. Softw. Sci. Tech. 1, pp.426–433(1989)
- [3] 大井, 安達, 夜久: Pascal から Hichart へのトランスレータの属性グラフ文法による記述と Prolog による実現, 電子情報通信学会技術研究報告, Vol.94, No.525, pp.89–96(1995)
- [4] Demers, A., Reps, T., and Teitelbaum, T.: Incremental evaluation for attribute grammars with application to syntax-directed editors, Conf. rec. the 8th Ann. Symp. POPL, Williamsburg, VA, Jan (1981)
- [5] 長野, 他: 木構造図用 CASE ツール間のデータ交換言語: DXL, 情報処理, Vol.35, No.4, pp.341–349(1994)