属性グラフ文法に基づいたHichartエディタ

1L－2

久保 知之† 安達 由洋† 安斉 公士†† 土田 賢省† 夜久 竹夫†††
†東洋大学 ††関東学園大学 †††日本大学

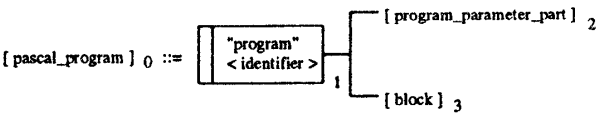
1．はじめに

階層型プログラム図式言語 Hichart [1] のためのプログラム図エディタを，Prologを用いて実現した．このエディタは，属性グラフ文法で定義されたPascal対応 Hichartの生成規則 [2] に基づく構文エディタ(syntax-directed editor)であり，生成や編集過程で構文エラーを引き起こすことはない．そして，レイアウト情報を含む属性をインクリメンタルに評価して Hichart図を動的に生成するという特徴を持つ．さらに，未定義の変数などの意味的エラーを検出し表示するなどの支援機能も実装している．

この Hichart エディタは，X Window と OFS/Motif を用いた GUI 上に実現されており，統合的ソフトウェア開発支援環境を実現するための重要な視覚的プログラミングツールとなる．

2．属性グラフ文法による生成規則の記述

今回開発した Hichart エディタの基本モデルとなる属性グラフ文法による Pascal 対応 Hichart の生成規則 [2] は図1の例のように定義される．




```
top(2) = TopY                    id(1) = 0
top(3) = bottom(2) + GapY        id(2) = 1
x(1) = RootX                     id(3) = id(2) + nc(2)
x(2) = x(1) + w(1) + GapX
x(3) = x(2)
y(1) = (y(2)+y(3)) / 2
w(1) = MinW
h(1) = get_height(["program",<identifier>])
cell(1) = "pre-defined_process"
string(1) = get_str(["program",<identifier>])
lines(1) = {get_line(1,[2,3])
```

図1　開始記号の生成規則

3．Hichartエディタの内部表現

図式を対象とする構文エディタの場合，終端記号だけでなく非終端記号も図形データとして表現できなければならない．さらに高度なエディタ機能を実現するためには，図形データも記号表現する必要がある．本システムでは，文献[2]のデータ表現を拡張し，Hichartの各セルデータを次のようなPrologの項として表現した．

```
extended_hichart_code(ID,Cell,location(X,Y),size(W,H),
    link(Parent,Children,Previous,Next),cellList([String]),
    Attr).
```

ここで，Cellは終端記号の場合にcell*n*（*n*は数値），非終端記号の場合にアトム nonterminal となる．

また，セルとセルを結ぶ接続線のデータは次のように表現される．

```
extended_connect_code(ID1,ID2,Linedata).
```

ただしLinedataにはセルID1とID2を結ぶ接続線の通る各点のX座標とY座標の値が[[x1,y1],[x2,y2],...]の形式で格納されている．

4．Hichartエディタの編集機能

本研究では，Hichartの構文エディタ実現のために，属性グラフ文法に基づきインクリメンタルな属性評価アルゴリズムを実装した．このアルゴリズムは，変数宣言がされていないなどの意味的な誤りを見つけるだけではなく，Hichartのプログラム図をインクリメンタルに配置するための属性評価も行う．

4．1　テンプレート，フレーズ，構文規則

Hichartのプログラム図は，セルと呼ばれるプログラムの制御と処理を表現する記号，セルを結ぶ線，そしてセルの内側と外側に記述される文字列であるフレーズによって構成される．

本エディタではこのセル，フレーズ，およびテンプレートをユーザが編集できる対象の基本要素とする．ここでテンプレートとは生成規則の右辺に対応したHichartの部分図であり，ユーザは非終端セルをテンプレートに置き換える操作を繰り返すことによりHichartプログラム図を生成していく．この操作は生成規則の適用に対応している．

A Syntax-directed Hichart Editor Based on an Attribute Graph Grammar

Tomoyuki KUBO †, Yoshihiro ADACHI †, Koushi ANZAI ††, Kensei TSUCHIDA †, and Takeo YAKU †††
† Faculty of Engineering, Toyo University
†† Faculty of Economics, Kanto Gakuen University
††† College of Humanities and Sciences, Nihon University

本エディタでは，効率のよい操作のために一連の生成規則の適用をひとまとめにしたものに対応する編集機能も提供している．さらに複数の置き換えが可能な場合は，それらの候補をポップアップウインドウで表示し選択できるような機能も提供している．

フレーズの入力はセル上にテキストウインドウを生成し，そのテキストウインドウを通して行われる．

4．2　挿入

柔軟なエディタを実現するにはテンプレートの置き換えだけではなく，挿入の機能も必要となる．

しかし，操作後のプログラム図が文法的に正しいことを保証するために，以下の(1)(2)(3)の手順で挿入を行う．

(1)挿入点の直前のセルが非終端セルになるまで生成規則を戻る．

(2)その非終端セルを展開する．

(3)一連の生成規則を再び適用する．

この操作は導出木の部分木の置き換えとしてみることもできる．

4．3　削除

削除も挿入と同様，文法的な正しさを保証するため属性グラフ文法に完全に基づいて実行され，文法の右側に対応する部分プログラム図を文法の左側に対応する部分プログラム図に置き換えることによって実行される．また本エディタの内部表現とPrologの持つ強力なパターンマッチング機能を用いることにより，削除のような操作も効率よく処理することができる．

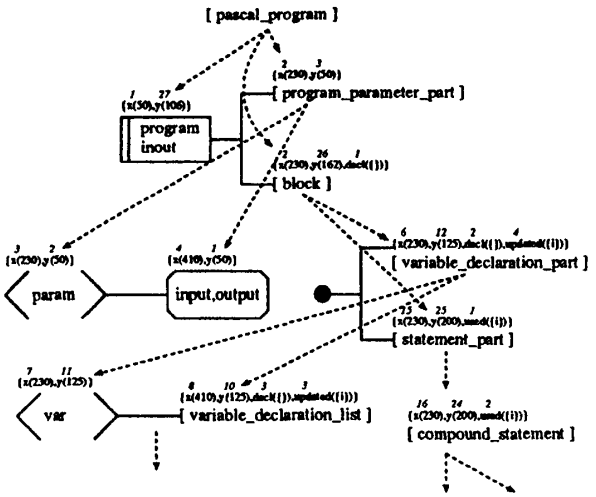


図２　Hichartの導出木とその属性の一部

4．4　属性評価

本エディタでは，未定義の変数などの意味的エラーを検出やHichartのプログラム図をインクリメンタルに配置するために，図２のような導出木を用いてインクリメンタルな属性の評価を行う．

5．Hichartプログラム図の作成例

本エディタは前節で述べたような操作を行うことにより任意のHichartプログラム図を書くことや任意のプログラム図から挿入や削除を繰り返し行うことにより他の任意のプログラム図に書き換えることもできる．

さらに本エディタが提供したコマンドをどのように用いてプログラム図を作成しても構文的に正しいもののみを作成する．図３はそれらの条件を満たしながら実際にHichartエディタを用いてプログラム図を作成した例の１つである．図３中の色の濃いセルが非終端セルである．

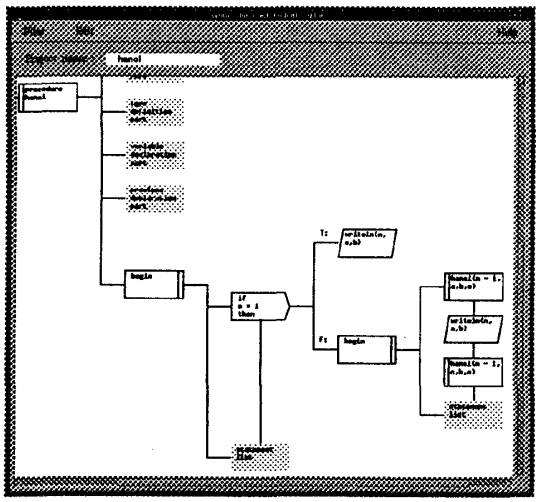


図３　プログラムの作成例

6．おわりに

属性グラフ文法で定義されたPascal対応Hichartの生成規則に基づく構文エディタをPrologにより実現した．

本エディタは，ある程度の柔軟性を持たせた編集機能を持ちながら，文法エラーを発生させない構文指向を実現したエディタとなっている．

参考文献

[1]夜久，他：階層型流れ図言語Hichartの情報処理記号，早稲田大学情報科学研究教育センター紀要，vol.3，pp.92-107(1986)

[2]大井，他：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，COMP94-100，pp.90-96(1995)