

属性順位グラフ文法によるプログラム図の構文解析

類瀬健二[†] 土田賢省[‡] 夜久竹夫[†]

[†] 日本大学文理学部応用数学科 [‡] 東洋大学工学部情報工学科

概要 木構造図一般を表現する中間言語として DXL(a Diagram eXchanged Language for tree-structured charts) が 1995 年に JIS X 0130 として制定されている。我々は Hichart を例に DXL に対応する木構造プログラム図に対する属性グラフ文法を考える。宮崎らにより, edNCE グラフ文法を基底グラフ文法とした DXL 対応 Hichart のための属性グラフ文法 HCGG(HiChart Graph Grammar) が定式化されている。

本論文では, HCGG に対して等価な順位グラフ文法を定式化し, その順位関係を用いた構文解析アルゴリズムおよび計算量の評価について考える。

Parsing of Program Diagrams with Attribute Precedence Graph Grammar

Kenji Ruise[†] Kensei Tsuchida[‡] Takeo Yaku[†]

[†]Department of Applied Mathematics, College of Humanities and Sciences, Nihon University.

[‡]Department of Information and Computer Sciences, Toyo University.

abstract DXL (a Diagram eXchanged Language for tree-structured charts) is an inner code for tree structured program diagrams and is registered in the Japan Industrial Standard as JIS X 0130. We consider Hichart as an example of program diagrams and parsing method an attribute graph grammar for tree structured diagrams with respect to DXL. Miyazaki introduced a HCGG (HiChart Graph Grammar) which is an attribute graph grammar with an underlying graph grammar based on edNCE graph grammar, for Hichart with respect to DXL.

In this paper, we formalize a precedence graph grammar which is equivalent to the HCGG at first. Next, we think a syntactic analysis algorithm with precedence relations and an appreciation of the complexity.

1 はじめに

プログラムの図表示は, 視覚的プログラミングツールやソフトウェア事例データベースにおけるプログラム情報の視覚化には不可欠な機能である。

近年, 様々なプログラム図の記述言語が報告され, それぞれに特徴を持っており, それらの言語に基づいた多くの CASE ツールが開発されている。なかでも, 1978 年に夜久らにより提案された Hichart(Hierarchical flowCHART description language)[1] は木フローチャートを基礎図式とする図形型言語であり, プログラムの階層構造, 制御の流れおよびデータ構造を同時に表示できるという他のプログラム図言語とは異なる特徴も持っており, 現在まで, Hichart 処理システムの開発が行われてきている [2], [3]。

一方, 木構造図データ交換言語 DXL(Diagram eXchange Language for tree structured charts)[4] は, 多種多様な木構造図に対する CASE ツール間でのデータを交換するための標準データ交換形式とし

て設計され, 1995 年に JIS X 0130 規格に制定された。各プログラム図言語に対して DXL との間の変換ツールを作成することにより, 異なるプログラム図言語の CASE ツール間でデータを相互に変換し, 利用が可能となる。現在まで, DXL に対応したシステムの開発はあまり行なわれておらず, DXL に対応したアプリケーションの開発は急務である。

DXL に対応した Hichart プログラム図生成のための属性グラフ文法 HCGG(HiChart Graph Grammar) を定式化されている [6]。HCGG は Rozenberg の edNCE グラフ文法 [5] を基底グラフ文法とし, 木構造図の初歩的描画条件のための属性規則を付加した属性グラフ文法である。基底グラフ文法を edNCE グラフ文法に定式化したことにより, プログラム仕様書 Hiform など基底グラフ文法を edNCE グラフ文法による表やシミュレータ等の他の処理系と統一的に扱える可能性がある [7]。

本論文では, HCGG に対して等価な順位グラフ文法を定式化し, その順位関係を用いた構文解析アルゴリズムについて議論する。

2 準備

この章では新しく構成する文法に用いるいくつかの文法、および Hichart の解説を行う。

2.1 edNCE グラフ文法 [5]

ここではグラフ文法の基本的な用語および edNCE グラフ文法の構成要素を定義する。

定義 2.1[5] Σ を頂点ラベル, Γ を辺ラベルとする。 Σ, Γ 上のグラフとは 3 項組 $H=(V,E,\varphi)$ である。ただし, V は頂点の有限集合, $E \subseteq \{(v,\gamma,w) \mid v,w \in V, v \neq w, \gamma \in \Gamma\}$ は辺の有限集合, そして $\varphi: V \rightarrow \Sigma$ は頂点をラベル付けする関数とする。□

定義 2.2[5] edNCE グラフ文法は次を満たす 6 項組 $GG=(\Sigma, \Delta, \Gamma, \Omega, P, S)$ であり, Σ は頂点ラベルの有限集合, Δ は終端頂点ラベルの有限集合, Γ は辺ラベルの有限集合, $\Omega \subseteq \Gamma$ は終端辺ラベルの有限集合, P は生成規則の有限集合, そして $S \in \Sigma - \Delta$ は開始記号である。ただし edNCE 文法の生成規則を $X \rightarrow (D,C)$ であらわし, X を非終端頂点ラベル, D をグラフ, そして $C \subseteq \Sigma \times \Gamma \times \Gamma \times V_D \times \{in, out\}$ を接続関係(connection relations) とする。 X を生成規則 p の左辺(left hand side), (D,C) を生成規則 p の右辺(right hand side) と呼ぶ。 □

2.2 属性 edNCE グラフ文法 [6]

ここでは, 属性 edNCE グラフ文法の構成要素を定義する。

定義 2.3[6] 属性 edNCE グラフ文法は次の条件を満たす 3 項組 $AGG=(GG,A,F)$ である。 $GG=(\Sigma, \Delta, \Gamma, \Omega, P, S)$ は基底グラフ文法, A は継承属性, 合成属性からなる属性集合, そして F は意味規則集合である。 □

2.3 順位グラフ文法 [8]

ここでは順位グラフ文法で用いる基本用語, 頂点間の順位関係, および頂点ラベル間の順位関係, そして順位グラフ文法の構成要素を定義する。

定義 2.4[8] V_G がグラフ G 内の頂点の集合, $E_G(v,w) \stackrel{\text{def}}{=} \{x \in \Gamma \mid (v,x,w) \in E_G, v,w \in V_G\}$ とする。この時, 二つの頂点のラベル組とは $lab_G(v,w) \stackrel{\text{def}}{=} (\varphi(v), E_G(v,w), E_G(w,v), \varphi(w))$ である。 □

定義 2.5[8] $\equiv, <, >, *$ は 2 頂点間の順位関係である。そのラベル間の順位関係 $R_{\bowtie} (\bowtie \in \{\equiv, <, >, *\})$ は導出可能なグラフ G が存在し, $v,w \in V_G$, v,w 間で有効な順位関係が存在するような $lab_G(v,w)$ のすべて

の集合である。 □

定義 2.6[8] 拡張辺ラベルの有限集合(extended edge label alphabet) は, $\Gamma_{\bowtie} \stackrel{\text{def}}{=} \Gamma \times \{\equiv, <, >, *\}$ である。 □

定義 2.7[8] 順位グラフ文法は 6 項組 $GG_{\bowtie}=(\Sigma, \Delta, \Gamma_{\bowtie}, \Omega_{\bowtie}, P, S)$ である。 $\Omega_{\bowtie}(\subseteq \Gamma_{\bowtie})$ は終端辺ラベルからなる拡張辺ラベルの有限集合である。 □
順位グラフ文法の構文解析は即時記述に対し, TOP を動作することにより行なわれる。

定義 2.8[8] 即時記述(Instantaneous Description)とは (G,K,Ψ) の 3 項組である。 G はグラフ, K はグラフ G 内の頂点の順序リスト, Ψ はそこまでの導出仕様の構成の集合である。 □

定義 2.9[8] $TOP(G,K) \stackrel{\text{def}}{=} G \setminus \{v_j, \dots, v_k\}$ (v_j から v_k までは等しい順位, $1 \leq j \leq k$) である。また $TOP(G,K)$ はシフトと還元と呼ばれる 2 つの型の動作(moves) からなる。 □

2.4 Hichart[1]

Hichart は 1978 年に夜久, 二木により発表された階層型流れ図言語である。Hichart は繰り返し記号 $\boxed{\quad}$ を初めて導入したプログラム図式言語で, プログラムを作成している要素間の制御の流れと階層的なつながりを疑似木構造グラフとして表現する。

2.5 HCGG[6]

DXL 対応 Hichart のための属性 edNCE グラフ文法を HCGG(HiChart Graph Grammar) と呼び, 生成規則 67 個, 意味規則 716 個からなる。

3 HCGG に対する順位グラフ文法

この章では, DXL 対応 Hichart のための属性順位グラフ文法を定式化する。

3.1 HCPGG

我々は HCGG に対して, 順位関係に矛盾が生じないように生成規則を追加, 編集し, Hichart 対応属性順位グラフ文法 HCPGG(HiChart Precedence Graph Grammar) を定式化する。

定義 3.1 HCPGG は $\langle GG_{\bowtie}, A, F \rangle$ で構成され, $GG_{\bowtie}=(\Sigma, \Delta, \Gamma_{\bowtie}, \Omega_{\bowtie}, P, S)$ は基底グラフ文法, A は属性集合, F は意味規則集合である。この文法は生成規則 69 個, 意味規則 728 個から構成される。 □

	[module_list]	"Profile"	[explanation]	[module]	"Identifier is"
[module_profile]					
"M_Packer"		R _≦			
[profile_module_list]					
[profile]	R _≧			R _≦	R _≦
[module_list]					
"Profile"	R _≧		R _≧	R _≧	R _≧
[explanation]					
[module]	R _≧			R _≦	R _≦
"Identifier is"	R _≧		R _≦	R _≧	R _≧

表 1: HCPGG の順位関係表

定理 3.2 L(HCGG) を生成する順位グラフ文法が存在する。

証明 HCPGG を考えると HCPGG の頂点間において、順位関係を矛盾なく定義できる。 □

表 1 は、HCPGG の順位関係表の一部であり、606 個の順位関係から構成され、2 頂点ラベル間はグラフにおいて直接連結される。

4 HCPGG に対する構文解析

この章では、HCPGG に対する構文解析アルゴリズム、およびそのアルゴリズムの計算時間について考察する。

4.1 HCPGG に対する構文解析アルゴリズム

HCPGG の構文解析アルゴリズムを、構文解析を行なう "HCPGG_analysis" と属性評価を行なう "attribute_evaluation" の 2 つから構成する。

4.1.1 HCPGG_analysis

"HCPGG_analysis" は入力グラフ、および入力グラフ内のひとつの頂点からなる順序リストを入力し、構文木を出力する。

Algorithm 1

```

HCPGG_analysis(Graph G, order list K){
  while(G != start graph){
    K = shift_order_list(G, K)
    p[j++] = find_production(K);
    G = replace_graph(G, K, p[j]);
    K' = rewriting_order_list(K, p[j]);
  }
  parse tree = generate_parse_tree(p);
}

```

アルゴリズム "HCPGG_analysis" では入力したグラフがスタートグラフになるまで "shift_order_list", "find_production", "replace_graph", そして "rewriting_order_list" の 4 つのステップを繰り返し、最後に generate_parse_tree" (492 行) で構文木を作成する。 "shift_order_list" (315 行) は、入力したグラフに対して順序ハンドルの探索を行ない、頂点を順序リストに追加する。 "find_production" (218 行) は順序リストから、順位ハンドルを右辺とする生成規則を発見する。 "replace_graph" (2774 行) はグラフの置き換え作業を行なう。 "rewriting_order_list" (250 行) は、グラフの置き換えに応じて順序リストから生成規則の右辺に相当する頂点を取り除き、左辺に相当する頂点を付け加える作業を行なう。そして generate_parse_tree" (492 行) は入力 p により構文木を作成する。ただし p は "find_production" で発見した生成規則の列である。

4.1.2 attribute_evaluation

"attribute_evaluation" は、構文木を入力とし属性評価を行なう。

Algorithm 2

```

attribute_evaluation(parse tree){
  S = attribute_parse_tree(parse tree);
  attribute_calculate(S);
}

```

アルゴリズム "attribute_evaluation" は、 "attribute_parse_tree" (115 行) において属性計算を行なう順番 (S) を求める。次に "attribute_calculate" (1267 行) は S に従って継承または合成属性の計算を行なう。

4.2 HCPGG による構文解析アルゴリズムの計算時間

定理 4.1 "HCPGG_analysis" は辺の数に対して線形時間で実行可能である。

証明 HCPGG において、グラフ内の辺の数が m の場合、そのグラフに対する導出列の長さは高々 m の定数倍となる。 "precedence_analysis" は、導出列の長さの回数だけ行なわれる。従って高々 m の定数倍行なわれる。 "shift_order_list" は "precedence_analysis" が行なわれている間に、順位ハンドルとなる頂点の移動を m+1 回、等しい順位ハンドルの発見を高々

入力	各頂点の頂点ラベル, 頂点番号,親の頂点番号, 最初の子の頂点番号, 2番目の子の頂点番号, 前の頂点番号,次の頂点番号 セルの幅,セルの高さ
出力	各頂点のx座標,各頂点のy座標 部分木の最下部のy座標 部分木の最上部のx座標 セルの識別番号 部分木内の存在するセルの数

表 2: 入出力データ

```

parse_tree[1104] Y = 4
parse_tree[1103] X = 19
parse_tree[1103] id = 1
parse_tree[1107] X = 16
parse_tree[1107] id = 2
parse_tree[1107] Y = 3
parse_tree[1107] bottom = 5
parse_tree[1106] Y = 3
parse_tree[1106] bottom = 5
parse_tree[1104] Y = 3
parse_tree[1104] bottom = 5
parse_tree[1101] Y = 0
parse_tree[1101] bottom = 2

```

図 1: HCPGG parser

m の定数倍行なっている。"find_production", "replace_graph", "rewriting_order_list" の計算時間は $O(1)$ である。また, "generate_parse_tree" の計算時間は導出列の長さの定数倍より $O(m)$ である。 □

定理 4.2 "attribute_evaluation" は線形時間で実行可能である。

証明 構文木の各々の頂点を高々2回しか通過しない。 □

5 HCPGG parser

この章では, HCPGG の parser の実行例を示す。入力形式はグラフで, 出力形式は属性評価を行なった構文木とし, データを表 2 に示す。この parser は C++ 言語で開発され, HCPGG.analysis は約 5000 行, attribute_evaluation は約 2000 行で構成した [10]。図 1 は HCPGG parser を実行したものである。

6 終わりに

本論文では, HCGG と等価な順位グラフ文法の存在を確認し, HCPGG を構成した。また HCPGG に対する構文解析アルゴリズムを構築し, 計算量の評価, Parser の開発 (約 7000 行) を行なった。

今後は Hichart 処理システムのエディタ等の開発および描画条件の改善を行っていく必要がある。

謝辞

本研究に関して助言をいただいた有田友和氏, 富山聖宣氏, 仲川俊一氏に深く感謝いたします。

参考文献

- [1] T.Yaku, K.Futatsugi, A.Adachi and E.Moriya; HICHART-A Hierarchical Flowchart Description Language-; *Proc. IEEE COMPSAC 11*, 157-163(1987)
- [2] 西野哲郎; 属性グラフ文法とその Hichart 型プログラム図式に対するエディタへの応用; コンピュータソフトウェア, Vol.5, No.2, 81-92 (1988)
- [3] 安達由洋, 大井裕一, 大澤優, 二木厚吉, 夜久竹夫; DXL 対応 Hichart プログラム図に対する属性グラフ文法; 日本大学文学部自然科学研究所研究紀要第 33 号, 149-164(1998)
- [4] 木構造図用データ交換言語 DXL; JIS X 0130-1995; 日本工業標準審議会, 1-36 (1995)
- [5] Grzegorz Rozenberg; Handbook of Graph Grammars and Computing by Graph Transformation; World Scientific(1996)
- [6] 宮崎征宏, 類瀬健二, 土田賢省, 夜久竹夫; プログラム図に対する描画を考慮した NCE 属性グラフ文法; 電子情報通信学会技術研究報告 Vol.100 No.52, 1-8(2000)
- [7] T.Arita, K.Tomiyama, K. Tsuchida, Y. Yaku et al; Syntactic Processing of Diagrams by Graph Grammars; *Proc. IFIP WCC ICS2000*, 145-151(2000).
- [8] Manfred Kaul; Practical Applications of Precedence Graph Grammars, Graph Grammars and Their Application to Computer Science, LNCS 291, 326-342(December-1986), Virginia: Springer-Verlag
- [9] 類瀬健二; 属性順位グラフによるプログラム図の構文解析; 日本大学大学院総合基礎科学研究科修士論文; <http://www.am.chs.nihon-u.ac.jp/~yaku/archive/thesis/6199M13/ronbun/paper.pdf>
- [10] 類瀬健二; hcpgg parser プログラム; <http://www.am.chs.nihon-u.ac.jp/~yaku/archive/thesis/6199M13/program/hcpgg-parser.cpp>