DXL 対応 Hichart の属性グラフ文法による定式化とトランスレータの実現*

5N－5

池上竜一† 東洋大学工学部§
安達由洋‡ 東洋大学工学部§
小倉耕一¶ 北海道東海大学‖
夜久竹夫** 日本大学文理学部††

1 はじめに

近年、視覚的プログラム開発支援システムに対する関心は急速に高まっており、実用化のための研究も報告されている。そこで提案されているプログラム構造図の記法は複数存在し、これらの異なる記法の構造図を相互に交換するために木構造図用データ交換言語の一つとして DXL が開発された。様々な表現形式を持つ木構造図は、この DXL を介して他の木構造図と相互変換することができる。

我々は今回、図 1 のように Hichart 図を他の木構造図と相互変換するために DXL と Hichart 間のトランスレータを実現した。この変換を行うために、DXL を Hichart 図に変換する生成規則を属性グラフ文法で記述し、それに基づき生成規則を Prolog で宣言的に記述した。また、Hichart から DXL への変換は、Hichart 内部表現を Prolog のパターンマッチ機能で構造解析することによって実現した。

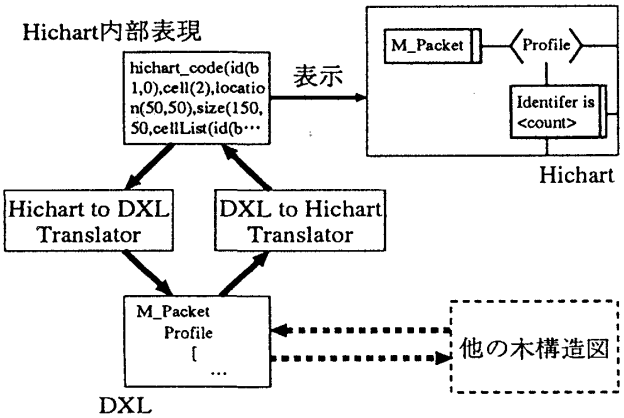

図 1. トランスレータの処理の流れ

2 DXL 対応 Hichart

2.1 Hichart とは

Hichart(HIerarchical flowCHART description language) は夜久等 [3] により提唱された階層型流れ図言語である。Hichart は繰り返し記号を初めて導入し、プログラムを構成している要素間の制御の流れと階層構造を疑似木構造グラフとして表現している。プログラムの階層構造とデータ構造が図表示に反映されるため、全体構造が把握しやすいことが特徴である。

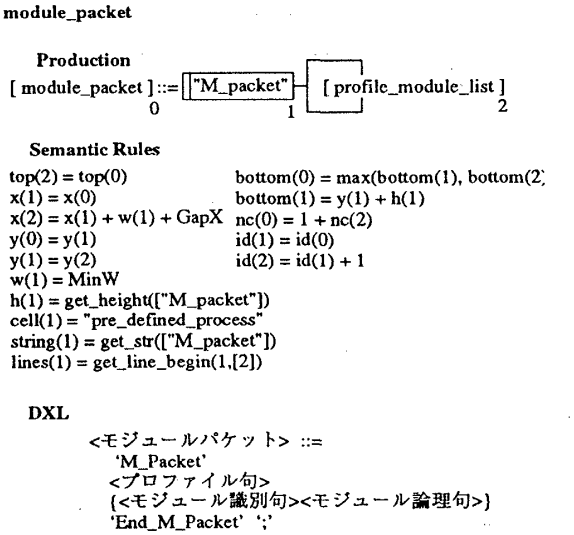

図 2. 属性グラフ文法の一部 (module_packet)

2.2 木構造型データ交換言語:DXL とは

DXL (Diagram eXchange Language for tree structed charts) は、日本工業規格などの場で標準化が進められている CASE ツール間のデータ交換形式の一つである。これは現在国内で使われている複数の木構造図間における設計情報の流通を目的として構築された言語である。

2.3 DXL 対応 Hichart 図生成のための属性グラフ文法

DXL に対応した Hichart 図を生成するために、その生成規則を属性グラフ文法を用いて図 2 のように定式化した。この文法は Hichart 図生成規則を文脈自

*A DXL-Hichart translator specified in an Attribute Graph Grammer and its Implementation †Ryuichi Ikegami ‡Yoshihiro Adachi §Faculty of Engineering, Toyo University ¶Koichi Ogura ‖Hokkaido Tokai University **Takeo Yaku ††Faculty of literature and science, Nihon University

由グラフ文法で、生成された図の配置情報を意味規則によって記述している。また、図の配置情報を導くための属性は、継承属性、合成属性とも、大井等[1] が Pascal対応 Hichartで定義したものと同様に定義した。

現在この文法は69の生成規則を持つ。Pascal の生成規則数の140と比べて少ないのは、DXL には Pascal におけるパラメータ宣言部にあたるものが存在せず、このため変数宣言などの変数処理に関する Hichart セルを生成する生成規則は全くなかったからである。

3 DXL に対応した Hichart トランスレータ

前節で定義した属性グラフ文法に基づき、DXL に対応する Hichart 内部表現への変換規則を属性文法により定義した。この属性文法は、 DXL 生成規則をもとに記述された文脈自由文法と、属性グラフ文法に基づく Hichart 内部表現を導出する意味規則からなる。この属性文法の属性と意味規則は、前節の属性グラフ文法の意味規則と同じものが用いられている。

3.1 DXL から Hichart へのトランスレータ

上述の属性文法を Prolog にて宣言的に記述することによって DXL ソースプログラムから Hichart 記号内部表現へのトランスレータ部を実現した。このトランスレータが DXL を Hichart 内部表現に変換するまでの流れは、以下のようである。

まず、ソースプログラムを入力して字句解析を行い、構文の解析に必要なデータを抽出する。次に、構文解析と属性評価を行い、解析木が求められる。この解析木から、 Hichart 内部表現を生成する。得られた内部表現は、 Hichart Viewer によって ワークステーション上に Hichart 図となって表示される。

3.2 Hichart から DXL へのトランスレータ

Hichart 内部表現を DXL ソースプログラムに変換するトランスレータは、次のように実現した。

Hichart 図のルートセルから、 DXL 対応 Hichart 図の生成規則をパターンマッチしていく。この時求められた Hichart 図の構造をもとに、対応する DXL プログラムを出力する。

この変換処理には Prolog の強力なパターンマッチ機能、すなわち Unification を利用している。

図3 に DXL とそれに対応した Hichart 図の表示画面を示す。

4 終わりに

木構造図用データ交換言語 DXL と Hichart を相互変換するトランスレータをワークステーション上に実現した。本トランスレータは変換部と表示部とで構成されており、変換部は IF/Prolog で約 3,000 行、表示部は IF/Prolog と OSF/Motif インターフェイスを用いて約 2,000 行で記述されている。このトランスレータが実現されたことにより、 Hichart 図と他の木構造図との間で、 DXL を中間表現として相互変換が可能になった。

今後、本研究の DXL 対応 Hichart 処理系を核にして、仕様記述も含む高度な視覚的プログラミング環境へと発展させていきたい。

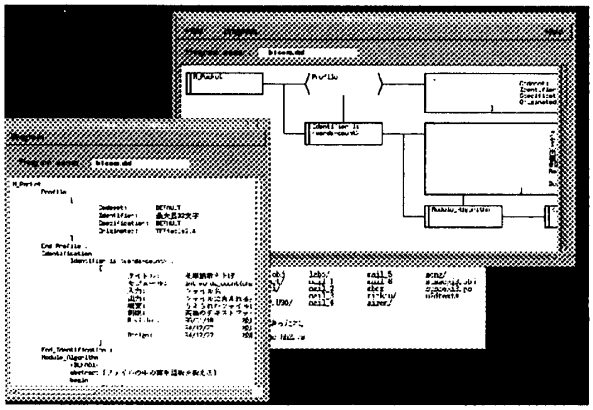

図 3. トランスレータの実行画面

参考文献

[1] 大井裕一, 安達由洋, 夜久竹夫: 属性グラフ文法に基づいた Pascal- Hichart トランスレータの Prolog による実現, INAP'95, pp.129-136 (1995)

[2] 長野宏宣, 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)

[3] Yaku, T., Futatsugi, K., Adachi, A. and Moriya, E. : HICHART- A Hierarchical Flowchart Description Language, *Proc. IEEE COMPSAC* 11, pp. 157-163 (1987) [4]