言語処理系クラスライブラリと一体化したオブジェクト指向構造エディタ

1L－1

一杉 裕志　平野 聡　須崎 有康　田沼 均　塚本 享治
電子技術総合研究所

1　はじめに

我々は、プログラミング言語の進化速度向上を目指し、言語処理系の部品のクラスライブラリ、 Lods を設計中である [2, 3]。このコンパイラキットの一部分として、構造エディタを実現するモジュールの基本設計を行なった。

構造エディタは、文や式などプログラミング言語の構文単位での挿入・削除を操作の基本とするエディタである。単純な文法エラーが早い段階で排除できる、インデントを自動的に行なってくれるなど、プログラムの生産性を向上させる特徴を多く持っている。 The synthesizer generator[1] など多くのシステムでは、属性文法による言語の定義を与えることにより、コンパイラと構造エディタを自動生成できるようにしている。

我々のコンパイラキットでは、拡張モジュールを追加していくことにより言語機能を拡張していけるものを目指している。しかし、従来のような文脈自由文法による記述では、シンタックスの拡張に制約が生じる。例えば、構文解析に影響を与えるような特殊文字や新しいキーワードを、言語仕様の他の部分に影響を与えずに追加することが不可能である。

そこで我々のコンパイラキットでは、言語シンタックスの拡張性を考慮に入れた構造エディタを、標準的プログラミング環境として提供することにした。このエディタの実現方法は、以下の特徴を持っている。

1. オブジェクト指向言語を用いて記述し、制御構造単位で表示方法、編集方法の変更・拡張が可能。
2. コンパイラキットと一体化しており、制御構造のシンタックスとセマンティックスをカプセル化した記述が可能。

2　拡張可能コンパイラキット Lods

我々が設計中の拡張可能コンパイラキット Lods[2] は、言語処理系作成者のための拡張性の高いクラスライブラリである。

ユーザプログラムは、言語処理系においてはノードと呼ぶオブジェクトの木として表される。これはプログラムの抽象構文木に相当する。木の各ノードは、 `:<N-default>` という名前のクラスを継承したクラスのインスタンスである。 if 文、 while 文、変数などをそれぞれノードとして表す。クラス `:<N-default>` には、すべてのノードが共通して持つ振舞いが定義され、各サブクラスには個々の構文の特徴に応じた振舞いが定義される。

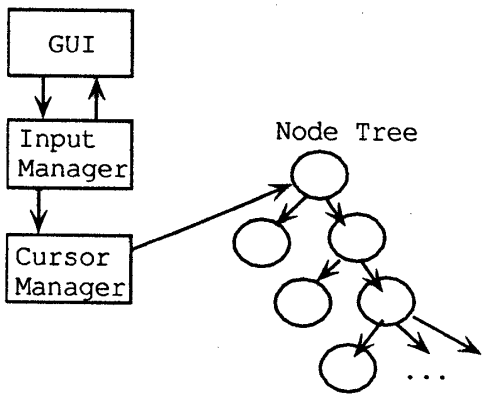

図 1: エディタの構造

このクラスライブラリは、拡張された継承機構を持つオブジェクト指向言語 Ld[3] を用いて記述している。この言語の機能により、新しい制御構造の追加、新しいデータ型の追加、新しいパスの追加などが、オリジナルの言語に対する差分モジュールとして記述できる。

3　エディタの構造

コンパイラキットのうち構造エディタを実現する部分は、言語のシンタックスに関係する部分とユーザインターフェースに関係する部分の2つに大きく分かれる。個々の部分も、実際には複数のクラス・メソッドから実現される。

編集時の各モジュールは、図1のような構造となる。ユーザのキー入力は、最初 input-manager というオブジェクトが受け取り、コマンドの種類によって、他のオブジェクトのメソッドを呼び出す。カーソル移動コマンドは、 cursor-manager というオブジェクトに、削除・挿入などの編集コマンドは、対象となるノードに処理をさせる。

cursor-manager は、編集中のプログラムを表すノードの木を管理する。現在カーソルがある位置のノードから木の root にあたるノードに至るまでの各ノードのリストと、そ

An Object-Oriented Structure Editor embodied in Programming Language Class Library. by Yuuji ICHISUGI, Satoshi HIRANO, Kuniyasu SUZAKI, Hitoshi TANUMA, Michiharu TUKAMOTO (Electrotechnical Laboratory).

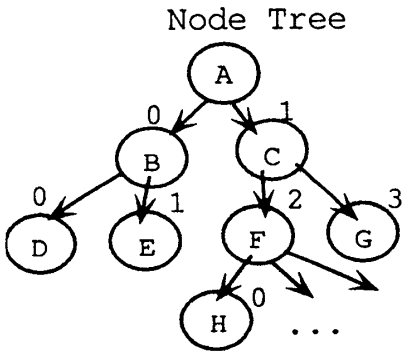

図 2: カーソルの位置の表現

れぞれが親ノードの何番目の引数かを表す数字のリストを保持する。例えば図2においてカーソルがノード H を指す場合は、ノードのリストは (F C A)、引数の位置のリストは (0 2 1) となる。カーソル移動コマンドが入力されると、これらのリストが更新される。

ノードの挿入・削除、ディスプレイへの表示内容の決定は、各ノードへのメソッド呼び出しによって行なう。編集コマンドが入力された場合は、対象となるノードに対して、「p 番目の引数の削除」、「p 番目の引数にノード N を挿入」などのメソッド呼び出しが行なわれる。そして、表示に変更があった部分に関しては、新しい表示内容を問い合わせるメソッドを呼び出し、表示させる。

4 個々の制御構造の定義

各制御構造のシンタックス、とりうる引数の数などは、制御構造の種類ごとに異なる。これらは、各制御構造を実現するクラスのメソッドとして定義し、コンパイル方法などと一緒にカプセル化する。

ノードの基本的な動作は、全ノード共通の親クラスで定義されているので、各制御構造のクラス定義部分ではいくつかの必要なパラメータを埋めるだけでよい。例えば、while 文を表すクラスでは、以下のメソッド定義によって表示内容と引数の数（可変個か固定個か）を指定する。

```
(method :display-data ()
  '("while (" X ") {" RET
    (list "   " X RET)
    "}"))
```

X は引数を表している。定義中、(list ...) とあるのは、その中を可変個繰り返してよいことを指定している。

なお、従来は言語を文脈自由文法で記述する際、引数としてとることができる非終端記号（文、式、シンボル、など）を指定するが、本システムでは、デフォルトでは任意のノードを引数として許す。とりうる引数の種類が限定される場合、言語の拡張性や、編集時の自由度が減るからである。本システムでは引数の種類のチェックは、コンパイル時に明示的に行なうことにしている。

5 モジュール間のインターフェース

各モジュール間のインターフェースはできる限り抽象化した。「プログラムはノードの木で表される」という点以外には、なるべく仮定をおかず、入力と表示の具体的な方法には依存しない。

例えば、各ノードに対するメソッド呼び出しのインターフェースは、「プログラムが1次元の文字列である」ということに依存していない。そのため、適当なメソッド定義をすることによって、

$$\sum_{i=1}^{10} f(i)$$

のように、1次元的な文字列という制約にとらわれてない構文も定義可能になる。

6 まとめ

言語処理系の部品のクラスライブラリの一部分として、構造エディタを実現するモジュールの基本設計を行なった。このエディタモジュールは、オブジェクト指向を用いた拡張性を意識した構造になっている。今後、いくつかの基本的な言語機能と拡張機能を実装していくことにより、本クラスライブラリを用いて構築した言語の言語機能とシンタックスの拡張性を実証していく予定である。

謝辞

本研究の一部はＲＷＣ計画の一環として「超並列システムアーキテクチャに関する研究」で行なわれたものである。関係各位に感謝いたします。

参考文献

[1] T.Reps., "Generating Language-Based Environments", MIT Press, Massachusetts 02142, 1984.

[2] 一杉裕志、平野聡、田沼均、須崎有康: "無制限に変更・拡張が可能なコンパイラの設計・実現方法の提案", 第４８回情報処理学会全国大会, 7G-4, March, 1994.

[3] 一杉裕志、平野聡、田沼均、須崎有康、塚本享治："拡張可能システムの記述を支援するオブジェクト指向言語", 日本ソフトウエア科学会第１１回大会, pp.29-32, 1994.