ソフトウエア製造ツールへの一提案

2D-7

ー言語“ＣＬＤ”のインプリメントと応用ー

小高　知宏°（早大・理工）　内山　明彦　（早大・理工）

１．はじめに

筆者らはプログラム部品の合成を支援するプログラム開発環境“A little interactive computing environment（Ａｌｉｃｅ）”の構築を行っている[1][2]。現在までにエディタ・データベースの試作を行い、また、プログラムの部品化をプログラム言語からサポートする言語“ Computer Language for Development（ＣＬＤ）”の設計を行った[3]。ＣＬＤでは部品の構成単位としてモジュールを設定し、部品の結合手段としてモジュール間をデータの流れで結ぶ「ストリーム」を導入した。また部品の再利用を容易にし、同時に開発の効率化をはかるためにモジュールの動的な結合を行うこととした。

今回はＣＬＤについて、試作したプロトタイプ処理系のインプリメントの方法と、ＣＬＤによるプログラミング例について報告する。

２．ＣＬＤのインプリメント

以下では、初めにＣＬＤのオブジェクト出力が実行される時の内部の挙動を説明し、次にＣＬＤコンパイラの構築方法を説明する。

a)実行時の内部挙動

先に述べたように、ＣＬＤは実行時に動的な制御を行うため、実行時インタプリタが必要である。実行時インタプリタはユーザが作成したモジュールの実行の制御とストリームの制御を行う。この際必要となるモジュール関係のデータは、ＣＬＤコンパイラによってモジュールテーブルに登録されている。またモジュール自体はコンパイルされたマシンコードであり、実行の制御やストリーム関係の処理のみが実行時インタプリタによって行われる。従って実行時インタプリタは、ストリームの状態の監視とモジュールの制御を行ない、必要に応じてモジュールテーブルを参照しながら起動すべきモジュールを決定することになる。（図１）すなわち実行時インタプリタは、モジュールテーブルに登録されているモジュールのうち実行可能（アクティブ）なものを捜し、順次これらに制御を渡す。またストリームの制御に必要な情報（ＣＬＤの世界内部の座標や入力ガードなど）をモジュールテーブルから検索し、あらたに起動するモジュールを決定する。

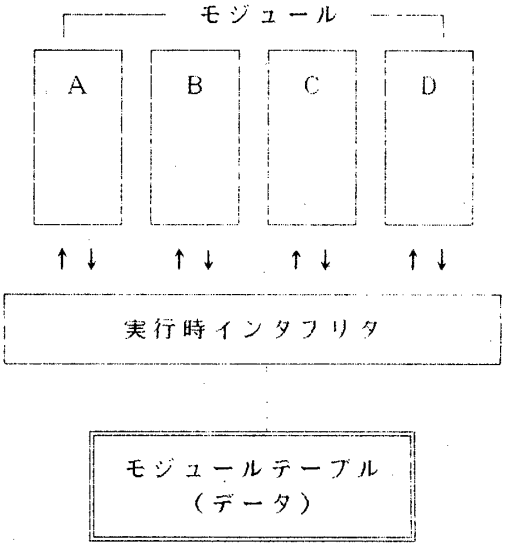

図1　ＣＬＤの内部構造

b)ＣＬＤコンパイラの構成

次に、上で述べたモジュール・実行時インタプリタ・モジュールテーブルを作成するＣＬＤコンパイラについて述べる。ＣＬＤコンパイラはＵＮＩＸのコンパイラコンパイラであるｙａｃｃ，字句解析プログラムｌｅｘ及びＣ言語によって記述されている。ＣＬＤコンパイラの受付けるソース言語はＣＬＤであり、コンパイラの出力はＣ言語である（次頁の図２）。すなわち、ＣＬＤコンパイラは中間言語にＣ言語を用いる。（あるいは、ＣＬＤを言語Ｃのプリプロセッサとして実現するとも言える。）

A Proposition on Software Tools Impremetation and Application of Language "CLD"

Tomohiro ODAKA, Akihiko UCHIYAMA
Waseda University

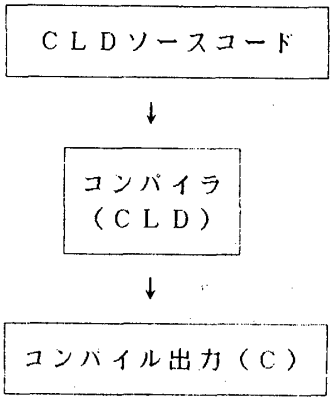

図2　ＣＬＤコンパイラ

３．ＣＬＤによるプログラミング

ごく簡単なプログラムの記述例を図３に示す。このプログラムはモジュールinputが端末よりコマンドを入力し、その他のモジュールがそれぞれの入力ガードに対応する入力を受け取るものである。これらのモジュールはストリームによって結合されるのみであり、モジュールの変更や追加は容易である。

```
module input
 interface
  ip for ip
 end_of_interface
 impremention
  ip()
   {
    char command[STR] ;
    char *gets() ;
    command[0] = 1 ;
    while(*command != 0)
    {
     printf("*** module input.¥n") ;
     gets(command) ;
     printf("    get command from keyboa
rd.¥n") ;
     ssend(command) ;
    }
   }
 end_of_impremention

module insert
 interface
  i for insertc
 end_of_interface

impremention
 insertc()
  {
   char command[STR] ;
   printf("*** module insert.¥n") ;
   sreceive(command) ;
   printf(" command is %s¥n",command) ;
  }
 end_of_impremention

module init
 interface
  init for initc
 end_of_interface
 impremention
  initc()
  {
   char command[STR] ;
   printf("*** module init.¥n") ;
   sreceive(command) ;
   printf(" command is %s¥n",command) ;
  }
 end_of_impremention
```

図３　ＣＬＤのソースプログラム例

４．まとめ

以上、ＣＬＤのインプリメントとプログラミングの方法について述べた。現在、ＣＬＤのソースプログラムを実際に処理させることで、プロトタイプ処理系の改良を行っている。今後はＣＬＤによるプログラミングをさらに進めるとともに、Ａｌｉｃｅシステムに ＣＬＤ処理系を統合する作業を行う予定である。

参考文献

[1] 小高　知宏　他　：　「ソフトウエア製造ツールへの一提案　開発支援環境Ａｌｉｃｅの設計と試作（第１報）」，電子通信学会研究会資料，EC84-66(1985年3月25日).

[2] 小高　知宏　他　：　「ソフトウエア製造ツールへの一提案　開発支援環境Ａｌｉｃｅの設計と試作」，情報処理学会全国大会予稿集　7F-10，(1985年9月12日).

[3] 小高　知宏　他　：　「ソフトウエア製造ツールへの一提案　開発支援環境“Ａｌｉｃｅ”の言語“ＣＬＤ”」，情報処理学会全国大会予稿集　6G-4，(1986年3月13日).