7D－5

図で記述されたMachine Descriptionを自動的にプログラムに変換するシステム

平原貴行+　山之上卓+　早田弘一+　安在弘幸++
+九州工業大学　++九州共立大学

1．はじめに

計算機の処理内容の記述方法の一つとして，図を用いこれを記述する方法が考えられる．図を用いることの利点として，システムの内容を直観的に把握しやすいという点がある．

本稿では，図で記述された Machine Discription を自動的にプログラムに変換するシステムについて論ずる．ここに示すシステムは，Abelson & Sussman[1] で示されたレジスタ計算機の概念に基づいている．

図はテキスト文字を使って記述する．このため，通常の EmacsやVi等で記述可能であり，また，何の変換もなしに，電子メールで配布したり，プリントアウトする事が可能である．

2．図によるMachine Discription

本システムは，データ路図と制御線図の２つの図を入力し，LispのＳ式で表されたプログラムに変換する[1]．

データ路図は，各レジスタおよび演算器の間のデータの流れを図示したものである．矢印上のｘ記号はボタンを示し，このボタンが押されたときのみ，その矢印上をデータが流れる．ボタンのない矢印には随時データが流れている．これらのボタンが正しい順序で押されないと計算機は正しく作動しない．その順序を示したのが制御線図である．また，制御線図では，レジスタ等の条件による分岐も示している。

本システムでは，これらの図はテキスト文字(+, -, |, (,), v, ^, x, etc) を用いて記述される．図は以下のように記述する．

- レジスタ，演算器，ボタン名，条件判断，ラベルは矩形で表す．
- レジスタ名等の名前には英数字（先頭は英字）のみをつかう．ただし，ボタン名のみ，矢印（(-, -)）を用いてもよい．
- 演算子の前には＊，ラベルの前には#，条件文の前には？を付けて表す．
- 制御図の先頭はラベル#startから始める．
- 条件分岐を表す矢印の１つに必ずｎ（no）またはelseをつける．
- 分岐先には必ずラベルを書く．ただし，条件分岐でｎまたはelseによる分岐先には書かなくてよい．

データ路図，制御線図の例を図１に示す．

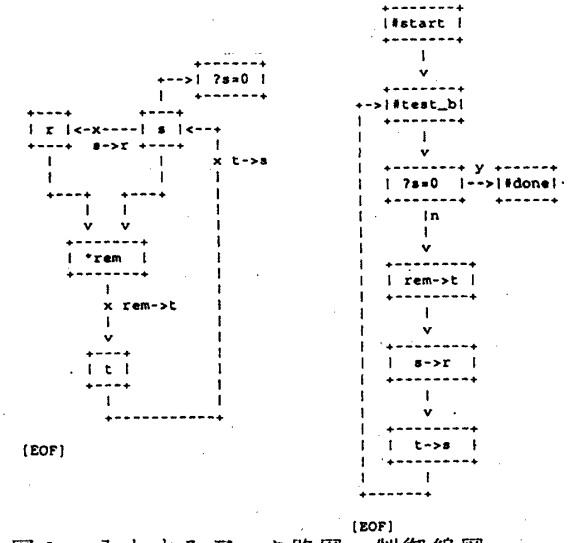

図１　入力するデータ路図，制御線図

A system which transforms a machine description described by figures into a program
Takayuki Hirahara+, Takashi Yamanoue+, Hirokazu Hayata+, Hiroyuki Anzai++
+Kyushu Institute of Technology, ++Kyushu Kyoritsu University

3．図の認識アルゴリズム

図の認識は以下のように行う[2]．[3]．

1. 図全体を走査して矢印の先端を検出しその位置を記憶する．
2. そこから矢印を逆にたどって始点を検出しその位置を記憶する．
3. 矢印にラベル（ボタン名および分岐の際の条件）が付いていればそれを読みとり記憶する．
4. 記憶された両端点の位置から両端の矩形を検出し，その内容を読みとり，記憶する．

１組の矢印および入出力矩形が認識されると続いてその入出力関係が記憶され，これによりプログラムへの変換準備が完了する．

認識の結果は，矩形の内容とその入出力（順序）を表す表に得られる．

4．プログラムへの変換

２章で作成された表をもとにして，プログラムの生成が行われる．

プログラムのデータ路を表す部分には，使用されているレジスタ，演算器および条件判断器とその入力元，また，存在すればボタンのラベル名が記述される．これらは，表から直接出力することができる．

しかし，制御を記述する部分では，記憶された制御列が正しい順番で並んでいるとは限らない（むしろ正しい順番でないケースが大半である）．そのため，これらを正しい順番に並べ変える必要がある．この並べ変えは，制御列の順序関係をたどることで実現できる．

```
(data-paths
 (registers
  ((name s)
   (buttons ((name t->s) (source (register t)))))
  ((name r)
   (buttons ((name s->r) (source (register s)))))
  ((name t)
   (buttons ((name rem->t) (source (operation *rem))))))

 (operations
  ((name *rem)
   (inputs (register r) (register s))
   (lisp-definition reminder)))

 (tests
  ((name ?s=0)
   (inputs (register s))
   (lisp-definition zero?))))

 (controller
  #test_b
    (branch ?s=0 #done)
    (rem->t)
    (s->r)
    (t->s)
    (goto #test_b)
  #done)
```

図２　図１より生成されたプログラム

条件分岐がある場合は次のようにする．分岐条件を順々に調べ，No，Else以外の分岐先をスタックに収めたのち，No，Elseの方向に進んで出力先をたどる．そのシーケンスが終わり（出力のない矩形あるいはすでに書き込み済みの制御）に達したらスタックを調べ，空でなければ取り出された分岐先から追跡を再開し，空であれば追跡を終了する．

こうして変換されたプログラムは，ファイル(*.spe)に出力される．図１のデータ路図，制御線図を入力した場合の出力例を図２に示す．

5．おわりに

本稿で論じた図表記の Machine Discription をプログラムに変換するシステムにより，簡単なハードウェア・シミュレータが実現できる．このシステムでは，言語表記の Machine Discription への変換にとどまったが，すぐに動かせるプログラムへの自動変換を実現することが今後の課題となる．

参考文献

[1] H. Abelson, G. J. Sussman, J. Sussman, Structure and Interpretation of Computer Programs, MIT Press (1985).

[2] T. Yamanoue, H. Hayata, H. Anzai, A Figure language for distributed system descriptions and its pre-compiler which can parse figures Proceedings ICSC'92: Second International Computer Science Conference, pp. 425-431 (1992).

[3] T. Yamanoue, H. Hayata, H, Anzai, A figure programming language for parallel supercomputers Transputers and Parallel Aplications, IOS Press, pp. 209-214 (1993).