3K－3

プログラミング教育支援システムのための知的エディタ

渡辺　弥寿夫　中沢　政幸　岩崎　光洋　小西　博之

金沢工業大学

１．はじめに

文献[1]でも指摘されているように，情報工学の教育では，理論(theory)，抽象化(abstraction)，設計(design)の３大原則に基づく理念が重要である．我々は，このような理念に基づくプログラミング教育の環境作りとして，計算機支援システムの研究を行ってきた[2,3]．今回は，ヒューマンインターフェイスの観点から学習者の記述のための編集環境を中心に述べる．

２．プログラミング

プログラミングを問題の理解，アルゴリズム開発，プログラム作成の３段階としてとらえ，学習者に正しいプログラムを作成する能力をつけさせるために，それぞれの段階で記述をするための環境及び検証系を構成した．

学習者のプログラミング開発を容易にするために各記述ごとに専用のエディタを設け，構文及び意味的な誤りがある場合は，それを指摘し，修正を行わせている．また検証における誤りの位置も画面上で表示できる．それぞれの環境を図１～３に示す．

３．エディタと検証

学習者が行うプログラミングの各過程の記述方法，それを記述する為のエディタの機能と，記述の検証について，それぞれの過程ごとに述べる．各検証系は，変数の属性として，ファイル，構造体，数値，文字列を扱うことができる．ファイルのマージ処理の問題[4]を例に，各過程の記述およびプログラム構造図を図５～７に示す．

３．１　問題記述

学習者はまず，入出力関係を示す変数リストを入力する．関係式の表現は，制限された日本語，関係式，算術演算式がある．また，図４のように，システムから与えられたことば（関数）を用いて問題の概略的な流れを視覚的に表現することができる．これを問題記述図と呼ぶ（図５）．概略的な流れを表現する方法として，手続き，関係式，入力，出力の記号を使う．また，問題記述図において頻繁に使われる式や関数を学習者自身で定義することもできる．

１）エディタ

問題記述図の構文的な誤り，未定義なパラメータ，誤った入出力関係はエディタにより即時指摘される．トランスレータは，問題記述図とその中の日本語を解析して関係式を導く．さらに，変数ごとに関係式をまとめ，中間コードとして，変数リスト，関数リスト，関係式を構成する．これを問題構造と呼ぶ．

```
関数名    関係式
open      file(?1,?a)&close(?a) -> file(?1,0)
close     file(?1,?a)&open(?a)&close(?b) -> file(?1,?b)
eof       f_num(?1,?a)&f_pt(?1,?b) -> ?b > ?a
```

図４　関数例

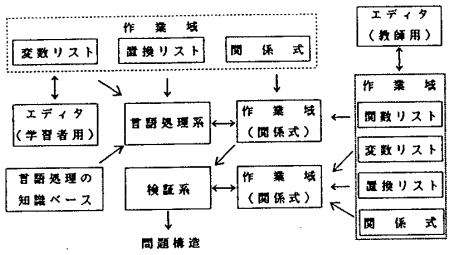

図１　問題記述の環境

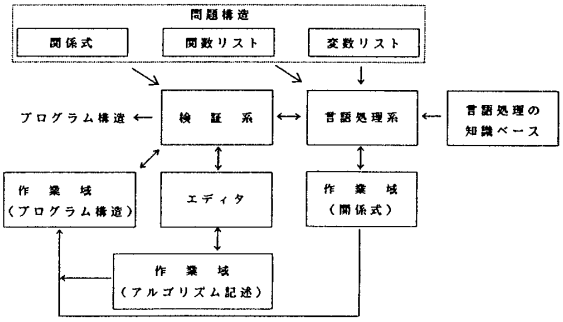

図２　アルゴリズム記述の環境

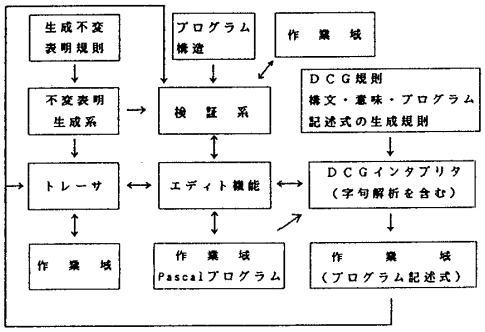

図３　プログラム記述の環境

２）問題記述の検証

問題構造をもとに学習者の変数を教師の変数と整合させて共通の変数リストを生成し，それをもとに教師の関係式と学習者の関係式を書き換えてマッチングにより検証を行う．

An Intelligent Editor for a Computer-Assisted Programming Training System
Yasuo WATANABE, Masayuki NAKAZAWA, Mituhiro IWASAKI and Hiroyuki KONISHI
Kanazawa Institute of Technology

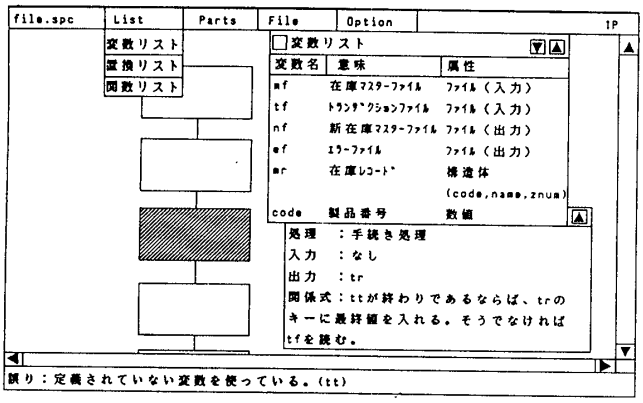

図５　問題記述例

３．２　アルゴリズム記述

学習者は，基本的な制御構造と制限された日本語によってアルゴリズムを記述する．設計の方法としては，段階的詳細化の方法に従う[2]．

１）エディタ

入力は，制御構造を選択し，日本語を入力する手順を繰り返すことにより行う．その際に，誤った制御記号によって不適切な制御構造とならないように制限している．また，日本語による記述が適切な記述であるかを日本語解析によって検査を行う．トランスレータは，制御構造の従属関係と日本語解析により関係式を導き，プログラム構造を生成する．

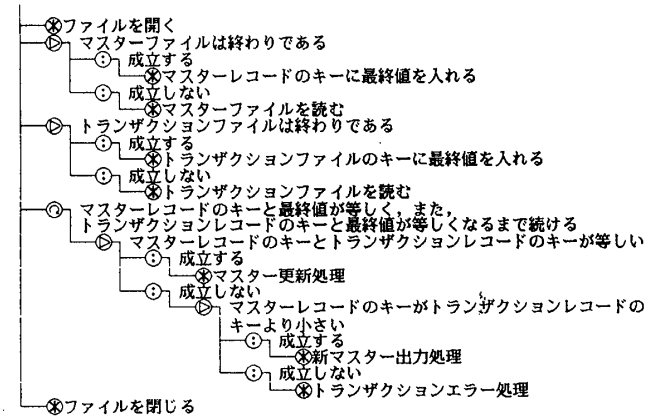

図６　アルゴリズム記述例

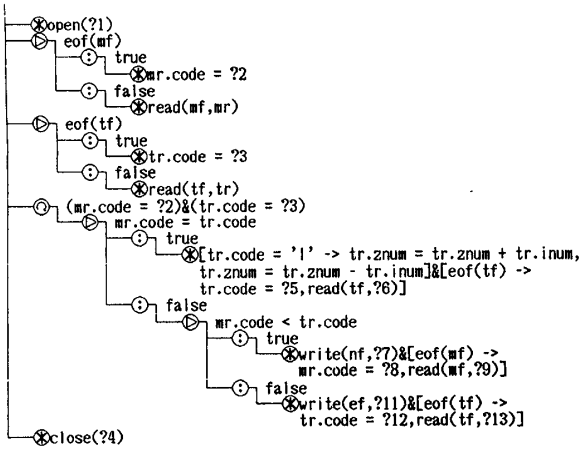

図７　プログラム構造図

２）アルゴリズムの検証

図７のようなプログラム構造図を基に得られた関係式の上下関係と，問題記述により得られた関係式を基に検証を行う．

３．３　プログラム記述

Pascalによってプログラムを記述し，トランスレータでは，論理的検証を行えるような制御構造を持った論理式であるプログラム記述論理式（中間コード）に変換する．そして，このプログラム記述論理式を用いて以下の処理を行う．

１）トレーサ

エディタ上で中間コードをもとに記号的に実行して変数の状態の表示とプログラムの流れをテキストに対応させて実行ステートメントを反転表示することでプログラムの動きを確認することができる（図８）．

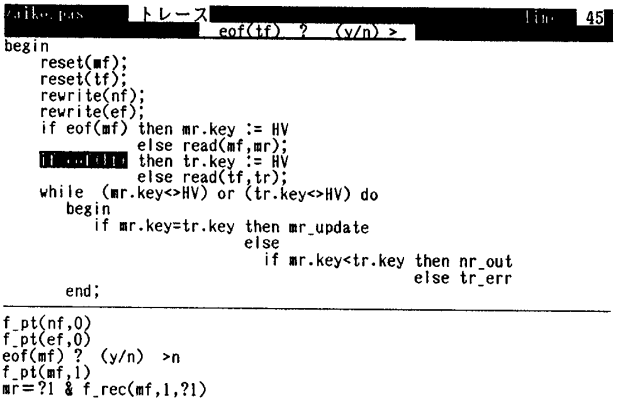

図８　トレース例

２）プログラム記述の検証

図７のようなプログラム構造図とプログラム記述論理式の制御構造が一致するか調べ，一致した部分でプログラム記述論理式から不変表明を求め，プログラム構造をもとに関係式の検証を行う．

４．おわりに

今回，専用の知的エディタを設けることによって，

・学習者にテキスト上で，エラーメッセージやその回復箇所などを指示できる．

・エディタを抜けて検証系を起動する必要がなくなった．

・問題記述エディタにおいては，図式化と自然言語による記述の併用によって，学習者の負担が少なくなった．

システムの実現は，現在Ｐｒｏｌｏｇと Ｃ言語を用いている．今回，構文・意味・内部表現生成の規則はＤＣＧによって，同時に記述したため，記述が長く複雑で理解しづらい．また，誤りの指摘も不十分な部分が多いので今後，これらを改良する必要がある．

参考文献

[1] P.J.Denning,et al. :"Computing as a Discipline," CACM, Vol.32, No.1,pp.9-23(1989).

[2] 中沢，渡辺，寺下：「プログラミング教育支援システムにおける段階的記述とその検証」日本科学教育学会研究会研究報告 Vol.3 No.2,pp.7-12(1988).

[3] 中沢，渡辺：「プログラミング教育支援システムにおける知識ベースと誤りのモデル化」電子情報通信学会研究報告 ET 88-10, pp.69-76(1989).

[4] 月刊情報処理試験,1988-9,pp.138-140.