3G-9　プロダクションルールを用いたプログラム作成支援システムに関する評価と検討

吉田　敦　田中　英彦　元岡　達

（東京大学工学部）

1．はじめに

ソフトウェアの生産性向上の為の有効な手段として、これまでに、さまざまな方法が考え出されてきた。そのなかの一つに、知識工学的手法を用いた、プログラミングの支援がある。この方法は、様々な応用領域への対応性や、優れたマン・マシンインタフェースの実現の可能性などの点で有望視されている。但し、実用的かつ汎用性のあるプログラムを対象としたものは、まだ、あまりない。[1]　我々は、以前、実用的で、ある程度汎用性のあるプログラムを対象としたプログラム作成支援システムの構成法について発表を行なってきた。[2]　その構成図は図1のとおりである。今回はこのうちのプログラム合成部を中心に、発表を行なう。

2．　システムの構成

本システムは、マン・マシンインタフェース、プログラム合成部、図形データ定義部、通信機構より成る。各々の要素の働きは、以下のとおりである。

1）　マン・マシンインタフェース・・・ユーザの入力を受け、解析して、行なうべき処理を決める。ユーザ固有の定義情報の管理も行なう。

2）　プログラム合成部・・・・プログラム一般の知識、ターゲット言語（FORTRAN）固有の知識、応用領域固有の知識を用いて、ユーザの与えた要求を対話的かつ段階的に詳細化し、目的とするプログラムを生成する。

3）　図形データ定義部・・・プログラムで用いるメニュー画面やシンボル図形等の設計を支援する。

4）　通信機構・・・・・・・他の計算機との情報交換や、ファイルの転送等を行なう。

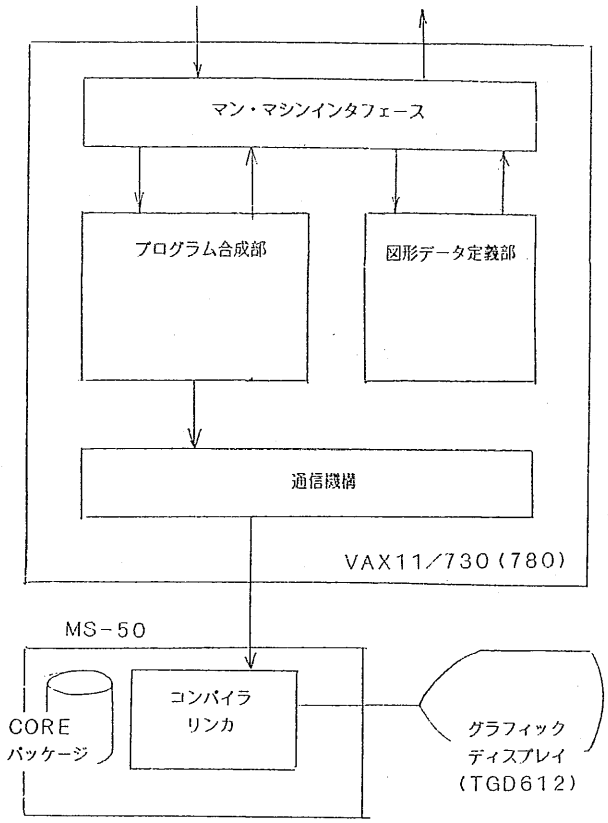

図1．　プログラム作成支援システム構成図

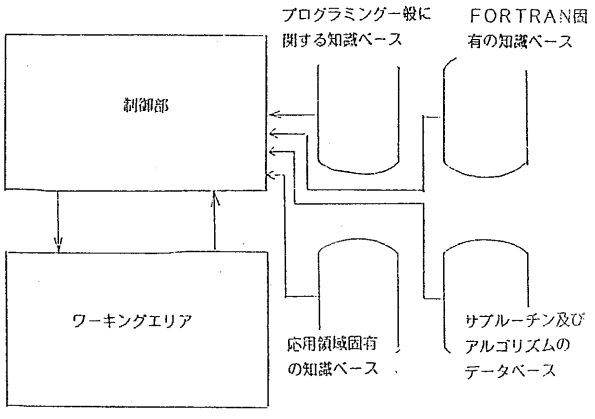

図2．プログラム合成部の構成

2.2　プログラム合成部の構成（図2）

プログラム合成部は、以下の要素から成る。

① 制御部　知識ベースの参照、ワーキングエリアへのアクセスなどの処理の制御を行なう。

② 知識ベース　以下の互いに独立した3つの知識ベースから成る

- プログラミング一般に関する知識
- 目標言語（FORTRAN）固有の知識
- 特定の応用領域（図形入出力）固有の知識

③ ワーキングエリア　目的のプログラムを合成するのに必要な情報を保持しておく。

④ サブルーチン及びアルゴリズムのデータベース

CORE（グラフィック・サブルーチンパッケージ）サブルーチン等の記述を持つ。

以上の各要素のうち①と②は、CProlog で記述されている。

2.3 知識の表現形式

システムの持つ知識は、プロダクション・ルールの形で表され、CProlog で記述すると、次のような形になる。（図3）

A：－B，！，C.

A・・・詳細化される要求

B・・・条件（付帯的情報）

C・・・詳細化の処理

このような形の知識（変換規則）を適用することにより、要求を段階的に詳細化し、目的とするプログラムを生成する。この方式の利点として、次のようなことが考えられる。

1） 部分的な決定、あともどりなどにより、単なるテーブル参照方式に比べ、柔軟な決定が可能である。

2） プログラム生成の規則が、説明や解析にも、利用できる。

3） 各知識の独立性が高いので、拡張性の高いシステムが構成可能である。

3. システムの動作

プログラム生成の手順は、次のとおりである。

① プログラムの制御構造に関する仕様を一定の形式（list形式）で入力。

② プログラムで用いるデータ構造の指定を、対話的に行なう。

③ 各知識を用いて、仕様を対話的、かつ段階的に詳細化。

④ プログラムをファイルに書き込む。

ユーザの入力から仕様を作りあげる部分や、マン・マシンインタフェースの部分は、まだ知識化されていない。（制御部の中に組み込まれている。）従って、ユーザは一定の形式と限られた語彙で仕様を記述する必要がある。

しかし、ユーザの入力した情報に不足があれば、システムがそれを検出し、ユーザへの、不足している情報についての問い合わせ、またはワーキングエリアに保持された付帯的情報からの推定により、不足している情報を補う。従って、ユーザは全ての情報を予め決めておく必要はない。

4. おわりに

対象とするプログラムに関する詳細情報やそれらの間の相互関連の取り扱いを、計算機側に負担させることは、従来の、パラメータ化技法などを用いたプログラム作成支援システムでも行なわれてきた。

しかし、詳細情報の取り扱いを、知識を用いて行なうことにより、従来のプログラム作成支援システムと等価な機能を持つものが容易に実現できることが明らかになった。

また、Prolog を用いることにより、ルールの選択や使い方に、柔軟性を持たせられることを確認した。

さらに、自然言語処理機能を持たせ、応用領域固有の知識の拡充を図ることにより、優れたマン・マシンインタフェースを持つプログラム作成支援システムが構成できることが期待できる。

現在、デバッグ支援の為のプログラムの解析機能の実装、及び、手続き等の選択の理由や、プログラムについての説明機能、先に述べた各知識の拡充が、当面、行なっていくべき仕事として残っている。

```
refinement([loop,Time,Process,end]):-
        var(nesting_level,N),
        variables(Vt,Time,int,Size),
      > !,
        refinement_loop(Vt,N,Size,Process).
refinement_loop(Vt,0,'[]',Process):-
        refinement2([loop,i,1,Vt]),
        set(nesting_level,1),
        implement(Process),
        refinement2([next,i]),
        set(nesting_level,0).
refinement_loop(Vt,1,S,Process):-
        length(S,1),
        !,
        refinement2([loop,j,1,Vt,'(i)']),
        set(nesting_level,2),
        implement(Process),
        refinement2([next,j]),
        set(nesting_level,1).
refinement_loop(Vt,2,S,Process):-
        length(S,2),
        !,
        refinement2([loop,k,1,Vt,'(j,i)']),
        set(nesting_level,3),
        implement(Process),
        refinement2([next,k]),
        set(nesting_level,2).
```

図3．知識の例 （プログラミング一般の知識）

参考文献

［1］ P. R. Cohen & E. A. Feigenbaum : The Handbook of Artificial Intelligence Vol2. Chap. 10 (Automatic Programming) 1982

［2］ 吉田、他：知識工学の手法を用いたソフトウェア作成支援システム 第27回情処全国大会 4B－9 昭和58年10月