『MRI-AI研究会会報 えいあい』no.12
(1989.3), 三菱総合研究所

自動プログラミング技術の現状

㈱三菱総合研究所
人工知能開発室　室長
玉　井　哲　雄

知識工学あるいはＡＩ技術は、エンジニアリング・医療・金融などの分野に応用の成果を挙げつつあるが、それらを実現しいてるソフトウェアの設計・開発過程そのものへの適用例は、現在のところあまり多いといえない。しかしソフトウェアの開発作業はすぐれて知識集約的であり、今後この分野へのＡＩ応用の展開が期待される。以下は、「自動プログラミング」という観点から、その動向を探ったものである。

1．自動プログラミングの種類

ソフトウェア開発にＡＩの技術を適用する試みには、多様なものがある。ここでは、とくにソフトウェアの設計・開発におけるＡＩ応用に焦点をあて、それを「自動プログラミング」と総称することにする。

このような意味での自動プログラミングには、多くの研究例や実践例があるが、それらを概観するために次の3種類に分類してみよう。

①形式的アプローチ
論理や数学的な形式性を備えた方法に基づき、自動的にプログラムを生成しようとするもの。ある意味でＡＩの本流に属し、比較的歴史も長い。

②知識的アプローチ
ソフトウェア設計や開発上の知識を知識ベース化して、ソフトウェア開発用のエキスパート・システムを作ろうとするもの。知識工学の隆盛とともに研究開発例が増えてきたもので、かなり新しい。

③実践的アプローチ
主として事務処理分野で、限定した問題領域におけるプログラムの自動生成や部品合成によるソフトウェア作成などの方法をとるもの。必ずしもＡＩ的なアプローチとはいえないが、第4世代言語やプログラム生成系などの形で実際に使われているものがかなりある。

C. Rich & R. Watersによる自動プログラミングについての示唆的なサーベイ〔1〕では、①エンドユーザ向け、②汎用的、③完全自動、という3つの条件をすべて満たす自動プログラミングの実現はまずむりだとし、実際には3条件のうちのいずれか1つをはずした次の3つのアプローチがありうるとしている。

・ボトムアップ：エンドユーザ向けという条件をはずし、プログラム・レベルから上向きの自動化を目指すもの。
・問題領域の限定：汎用性をあきらめ、分野を限定して自動化を目指すもの。
・アシスタント：完全な自動化をあきらめ、ソフトウェアの設計や開発の助手としての役割を果たすシステムを目指すもの。

この考え方と上記の分類とは視点が異なるが、形式的アプローチはボトムアップと、知識的アプローチはアシスタントと、実践的アプローチは領域限定と、重なる部分が多い。

2．形式的アプローチ

形式的アプローチの特徴は、次のようである。

・形式的仕様を前提とする。仕様記述方法の研究自身も、含まれる。
・理論的に明確な基盤をもち、小規模プログラムに対しては自動合成能力が実証されている。
・研究的な色彩が強く、実用的なレベルにはまだ達していてない。

主な方法としては、論理に基づく演繹的な手法と、プログラム変換法がある。代表的な事例を次表に示す。

システム名	開発者	発表年	特徴
PROW	Waldinger & Lee	1969	定理証明に基づく演繹的アプローチの原理を示したもっとも初期のもの。
unfold/fold system	Burstall & Darlington	1977	変換法のもっとも重要な原理を示したもの。後に、これを機械化したシステムが種々作られた。対象は関数型プログラム。
DEDALUS	Manna & Waldinger	1979	多くの個別的な変換規則の集合でプログラム変換を行おうとするもの。
PSI, CHI	C. Green 他	1976	プログラム生成部分は変換法に基づいているが、ソフトウェア開発の全体的な支援をねらったもの。知識処理的な色彩も濃い。
CIP	F. Bauer 他	1981	仕様から実行レベルまで一括して記述できる広範囲言語と、その上での変換システム。
ϕ_0, ϕ NIX	D. Barstow	1982	問題分野を限定したプログラムの自動合成系。知識型に近いともいえる。
SAFE	R. Balzer 他	1985	GISTという仕様言語から、プログラム生成を目指すもの。
Logic Program Synthesis	佐藤、玉木	1984	論理型プログラミング言語に対する変換法の適用。
PX	林、中山	1987	PXという論理システムにおける証明からプログラムを生成しようとするもの。

3．知識的アプローチ

知識的アプローチの特徴は、次のようである。

・他分野で成功例のある知識ベースを用いたエキスパート・システムを、ソフトウェア設計および開発のフェーズに適用するものである。

・形式的アプローチと比べると、プログラム生成よりも設計作業の支援を目指すものが多い。またそれに関連し、完全な自動化よりも人間の補助的役割をねらっている。

・ソフトウェア開発過程における生成物（仕様、プログラム、テストケース、データ、文書、など）をオブジェクト・ベースとしてしまい、それらに対するプロセスについての手順、条件、評価などの知識を、たとえばルール・ベースとして蓄え利用するというような、「知的プログラミング環境」あるいは「ソフトウェア・エンジニアリング・データベース」ともいうべき概念を核とするのが典型的である。

・まだ概念的な段階の研究例が多く、その概念を確かめるためのプロトタイプが作られているというのが、実態である。

代表的な例を次表に示す。

システム名	開発者	発表年	特徴
Programmer's Apprentice	C. Rich 他	1978	大規模プログラムの開発に役立つようなプログラマの助手を勤めるシステムの研究開発を目指すもの。基本的なプランを知的インターフェースを介して検索・利用する。
IDeA	Lubars & Harandi	1986	データフロー図による設計に際し、抽象的なスキーマを具体化したり、詳細化したりする過程を支援する。
SIES	Tsai & Ridge	1988	データフロー図で与えられた要求仕様から構造チャートを導出するのを支援する。
ASPIS	P. Puncello 他	1988	SADTの記法をベースに、システム分析と設計の知識をもったアシスタントの開発を目指す。
Kate	Fickas & Nagarajan	1988	ソフトウェアの目標戦略と具体策の関係を表すモデルを利用して、目標に照らし作られた仕様を評価し、問題点を指摘する。
PSDL	Luqi & Ketabchi	1988	主としてリアルタイム用のプロトタイプ記述言語と設計の再利用の概念をシステム化することにより、プロトタイピング・アプローチによる設計を支援する。
KIPS	杉山、他	1984	日本語によるシステム記述を意味ネットを用いて解析し、HyperCobolによるプログラムを生成する。
ICAS-REUSE	千吉良、他	1987	設計仕様の再利用を目指し、仕様のライブラリ化と検索、冗長度のチェックや不足情報の追加などの支援を行う。

4．実践的アプローチ

ここにあげる実践的アプローチの特徴は、次のようである。

・第4世代言語、プログラム生成系、部品化・再利用、などのツールや方法との親和性が高い。またCASE（のとくに下流工程を対象とするもの）という位置づけを与えられているものもある。

・問題領域としては、事務処理分野を対象にするものが多いが、なかにはシステム・ソフトウェアを対象にするものもある。

・開発プロセスを支援するタイプのものと、プログラムの自動生成に焦点をおくものとがある。

いくつかの例を、次表に示す。

システム名	開発者	発表年	特徴
Marvel	G. Kaiser、他	1988	Cプログラムを対象とし、プログラム、モジュール、コンポーネント、などに関する情報を収めたデータベースと、それらを処理するプロセスに関するルールを収めたルールベースを用いて、ソフトウェア開発を支援する。
CASE/MVS	A. Symonds	1988	大規模ＯＳソフトウェアを対象に、そのモジュールなどの性質をオブジェクト・ベースとして蓄積し、その静的特性や動的特性を解析する。またそれからプログラムを生成することも、目指されている。
SPACE	原田	1987	決定表に類似した形式による仕様から、Cobolプログラムを生成する。
PAPS	古宮	1987	プログラムのスケルトンや部品のライブラリを利用しCobolプログラムを生成する。
PRECOBOL	大木	1987	オブジェクト指向の概念に基づき、部品利用によるプログラム作成を支援する。

参考文献

〔1〕Rich, C. and Waters, R. C., "Automatic Programming : Myths and Prospects," IEEE Computer, August 1988, pp. 40-51.

〔2〕玉井哲雄、"ソフトウェア工学と知識ベース"、電子情報通信学会誌、vol. 71(1988), No. 9, pp. 915-918.