3H－3

入出力例題からの生成検査法プログラム合成手法の提案

井上義史　永田守男
慶應義塾大学　理工学部

1．はじめに

近年、ソフトウエアの需要が増え、プログラムをコンピュータによって自動合成する手法が開発されてきた。しかし、実際に生成できるプログラムは、アルゴリズムの型が限られていたり[1]、初歩的なものに限られたりしたままである。

そこで、本研究では、この種の問題で取り上げられることの少なかった「生成検査法」と呼ばれるプログラムを対象として、入出力例題から自動合成する手法を提案する。ここでの提案の特徴は、入出力例題の入力引数と出力引数の関係から用意したライブラリ中の生成系述語を抽出し、出力引数どうしの共通情報から検査系述語を絞り込んだことである。

2．生成検査法について

「生成検査法」とは、確定的なアルゴリズムのない場合に論理型言語の非決定性を有効に使って作られるプログラム構造（図1)で、近年盛んに研究されている遺伝子アルゴリズム[2]などもその一つである。また、この方法は、非決定性を持つ論理型言語での実現に適している。

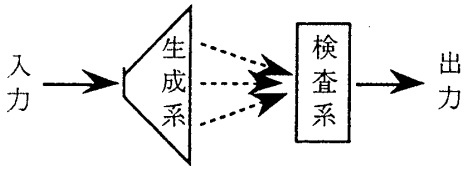

図1．生成検査法の構造

この構造を本研究の対象とする言語Prologで表すと

```
find(X) :- generate(X), test(X).
```

となる。これはgenerateで仮に生成したXをtestで検査し、この検査に通ったものが求める答になるという意味である。

3．自動合成上の問題点

このようなプログラムを合成するには、生成系(generate(X))と検査系(test(X))の述語を確定できればよい。このとき、次の3点が問題になる。

- 単純な述語どうしの組合せであるか否か、つまり生成系に含まれる述語と検査系に含まれる述語だけからなる単純なプログラムかどうか。
- その単純な場合に生成系や検査系にくるべき述語の候補をどうやって決定するか。
- 生成系や検査系の前後に補助述語が入る複雑な場合はどのように処理するか。

4．検討

以上の問題点を解決するために、リスト処理に関する既存の生成検査法プログラムを分析した。

その結果、次のことが判明した。

- 生成系にくる可能性の高い述語
  - * 順列・組合せに関する述語(G0)
  - * データベースを参照して候補を抽出する述語(G1)
- 検査系にくる可能性の高い述語
  - * リストの処理を行う基本的な述語(T0)
  - * データベースと照合して検査する述語(T1)
  - * 再帰構造をもつ述語(T2)
- 生成系の前または検査系の後にくる補助述語
  - * リストの処理を行う基本的な述語(A)
- 最も単純な場合(G0-T0型)の特徴
  - * 入力例、出力例ともに平坦リストで、出力例の全ての要素が入力例のリストに含まれている。

以上の分析を基に合成アルゴリズムを作成した。

A Proposal of a method for synthesizing generate-and-test programs from input/output examples
Yoshifumi INOUE, Morio NAGATA
Keio University

5．合成アルゴリズム

概略は、図２のとおりである。

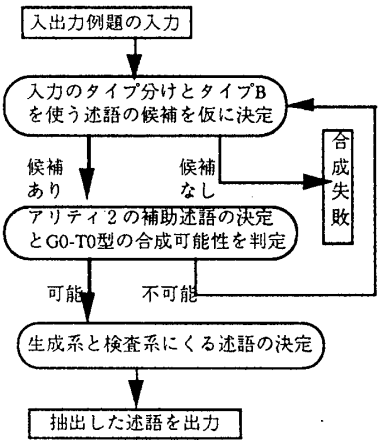

図２．自動合成の流れ

まず、入出力例題中の入力と出力を比較して、生成されるものの原料となる入力（タイプA）と生成方法や検査方法などを指定している入力（タイプB）に分け、タイプBを使う述語の候補をライブラリから仮に抽出する。

次に、タイプAと出力のデータ型がともにリストになっている時以外は、それらをリストに変形する補助述語を決定する。そして、変形後のリストにおいて、タイプAのリストが出力リストの要素を全て含んでいる場合、G0-T0型の合成が可能であると判定する。合成が可能と判断されたときは、タイプAと出力間に成り立つ関係により生成系述語を、出力だけの特徴により検査系述語をライブラリから抽出し、述語を並べ替え、出力して終了する。一方、合成が不可能と判定されたときは、入力のタイプを分けるところからやり直す。そして、仮決めで候補となる述語がなくなれば、合成に失敗ということで終了する。

6．実験とその結果

システムを試作し、Prolog教科書5冊の例題[3,4,5,6,7]を使い、評価実験を行った。システムへの入出力例題の前提は、次の２点である。

・入出力例は、アトムまたは空でないリスト。

・例題のアリティは２か３で、一つ以上の入力と一つの出力をもつ。

本システムの対象範囲{G0, T0, A}での実験結果は、次のとおりである(表１)。

プログラムのタイプ	本数(%)	意図通りに合成	意図と異なる合成	合成失敗
F :- G0	14 (29)	14	0	0
F :- G0, A	3 (6)	1	2	0
F :- A, G0	2 (4)	1	0	1
F :- G0, T0	10 (20)	7	3	0
F :- G0, TO, A	4 (8)	0	4	0
F :- A, G0, T0	3 (6)	2	1	0
G1, T1, T2混合形	13 (27)	0	5	8
合計	49 (100)	25	15	9

注）Fは、目的プログラムの頭部を表わす

表１．実験結果

7．評価と結論

合成率が極めて高かったことから、ライブラリ述語の与え方、入出力例題間の特徴を用いた推論に妥当性があったといえよう。

しかし、システムの成長性と目的プログラムの正誤に関する問題が残っているので、実用化に向けて別の次元でのアプローチが必要である。

<参考文献>

1) 小泉昌紀,永田守男: 類推を用いた入出力例題からの論理プログラムの合成手法の提案, 情報処理学会論文誌, Vol.32, No.9, pp.1080-1089 (1991).

2) 北野宏明 : 遺伝的アルゴリズム, 人工知能学会, Vol.7, No.1, pp.26-37 (1992).

3) L.Sterling,E.Shapiro : The Art of Prolog, The MIT Press (1986).

4) W.F.Clocksin,C.S.Mellish : Programing in Prolog, Springer-Verlag (1981).

5) 古川康一 : Prolog入門, オーム社 (1986).

6) 新田克己,佐藤泰介: Prolog, 昭晃堂 (1986).

7) A.L.Johansson,A.Eriksson-Granskog : Prolog versus You, Springer-Verlag (1989).