TRSプログラムの自動合成システム

1J-7

菊地俊彰　侯本慧　富樫敦　野口正一
東北大学電気通信研究所

1. はじめに

人工知能、あるいはソフトウエア工学(科学)の目的の一つに自動プログラミングがある。

自動プログラミングのアプローチには、完全仕様からプログラムを生成する方法と、例題などのような不完全な仕様から生成する方法とがある。前者は演繹的推論[4]やプログラム変換によるもの[2]で、後者は例からのプログラミングまたは帰納的推論によるもの[1,5,7]である。

本論文では、項書き換え系(Term Rewiting System)を対象とした例によるプログラミングを考える。具体的には、

plus(0,3) = 3
plus(1,4) = 5

のような関数の入出力の例を与えたとき、

plus(0,Y) ▷ Y
plus(s(X),Y) ▷ s(plus(X,Y))

のようなTRSプログラムを合成するようなシステムである。

2. 合成システム

本論文の目的は、入出力の例から帰納的推論によってTRSプログラムを合成するシステムの作成ある。基本となるアルゴリズムは、Shapiroがモデル推論で用いた方法に基づく。つまりTRSのルール全体の枚挙を考え、読み込んだ事実に対し正しいルールを拾い上げ、間違っているルールを捨てるという方法である。

2.1 システムの構成

システムの主要部は、次の3つからなる。

- インタープリター
- デバッガー
- ルールジェネレーター

2.2 インタープリター

与えられた項を現在持っているルールに従って簡約する。プログラムを合成するときは、入力例を受け取って簡約を行う。

2.3 デバッガー

入力例の簡約結果と出力例とを比較する。両者が一致しないときはルール(プログラム)にバグがあるので必要な操作を行う。そのバグの種類とそれぞれの操作は次の通りである。

i) プログラムが停止しない。
入力例の簡約がある定められた有限時間内に終了しない場合。次の例を試みる。

ii) 答えが期待する正規形にならない。
入力例の簡約結果に(あらかじめ定めた)正規形に含まれない関数記号が入っている場合。ルールが足りないのでルールジェネレーターに新しいルールを作ってもらう。

iii) 答えが間違っている。
入力例の簡約結果が正規形にはなるが、出力例と違っている場合。(プログラムに)間違っているルールが含まれているので、神託(oracle)を基に簡約木(reduction tree)を探索する。その結果、間違っているルールが見つかったら、削除する。

2.4 ルールジェネレーター

要求に応じてルールを一つずつ生成する。その方法は精密化演算子を用いた枚挙である。関数名と引数の数とから最も簡単な一般形のルールを作り、精密化によってより複雑な形にする。しかしそれだけでは停止性を満たすことは難しいので、orderingを導入する。つまり、初期条件として関数記号の順序を与え、再帰的経路順序(recursive path ordering)や辞書式経路順序(lexicografic patn ordering)によってルールの両辺の複雑さを計算し、(複雑さが)より小さいtermに書き換えないものは廃棄する。

2.5 神託(oracle)

神託は、本質的には入出力例と同じである。(神

Synthesis System for TRS Programs
Toshiaki KIKUCHI, Ben Hui HOU, Atsushi TOGASHI, Shoichi NOGUCHI
Research Institute of Electrical Communication, Tohoku University

託は、「この項を簡約すると答えは何か」という形で出される)しかし、後者は予めユーザーが入力するのに対し、前者は必要なときシステムの要求に応じて入力される。

2.6 アルゴリズム

① n番目の入出力例を読み込む。
② $i(1 \leqq i \leqq n)$番目の例についてインタープリターによって入力例を簡約する。
③ その結果と出力例とをデバッガーに渡し、合っているかどうか調べる。
④ 合っているときは$i+1$番目の例に進む。
⑤ そうでなければ、デバッガーでバグの種類を調べ必要な操作を行う。$1 \sim i-1$番目の例について②~④の手続きを繰り返す。
⑥ $i=n$になるまで②~⑤を繰り返す。
⑦ $n+1$番目の入出力例を読み込む。

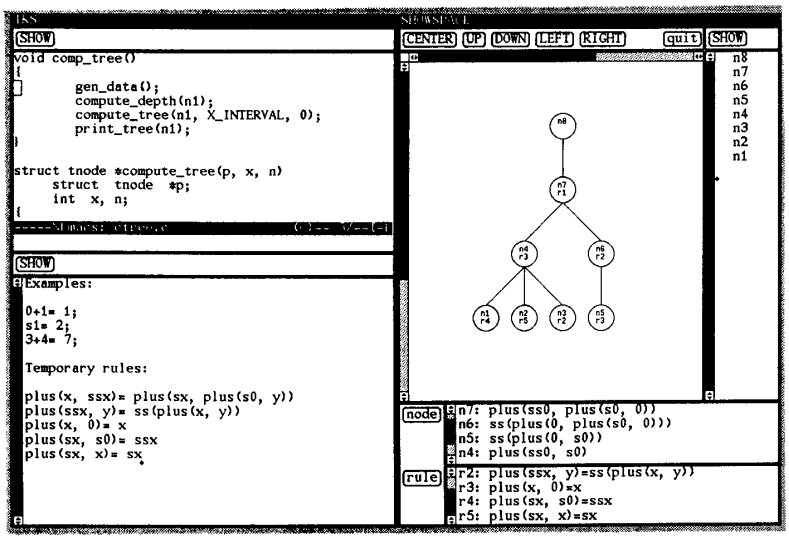
3. ウインドウシステムを用いたインターフェース

本システムは、KCLを用いてインプリメントされているため、UNIX上で動く。しかし、そのままではあまりインターフェースが良いとは言えないので、SunView上でのインターフェースを考える。

下図において、左側のウインドウは導出木(reduction tree)を表示する。右側はC-shellとtext editorが走っていて、システム本体を動かす。

4. まとめ

KCL及びSunViewを用いてTRS合成システムを作成した。



参考文献

[1]Biermamm, A. W. "The inference of regular Lisp programs from examples", IEEE Transactions on Systems, Man and Cybernetics SMC-8, pp 585-600, 1987.

[2]Darlington, J. "Program Trasformation", Functional Programming and its Application, Cambridge Univ. Press, pp 193-215, 1982.

[3]Huet, G. and Oppen, D. C. "Equations and Rewrite rules : a Survey", Formal Languages : Perspectives and Open Problems, Ed. Books R, Academic Press, pp 349-393, 1980.

[4]Manna, Z., Waldinger, R., A Deductive Approach to Program Synthesis, TOPLAS, Vol. 2, No. 1, pp 90-121, 1980.

[5]Shapiro, E. Y. "Inductive Inference of Theories From Facts", Tecnical Reports 192, Yale Univ., 1981. (有川節夫訳「知識の帰納的推論」, 共立出版, 1986)

[6]Shapiro, E. Y. "Algorithmic Program Debugging", Ph. D. Thesis, The MIT Press, 1982.

[7]Summers, P. D. "A methodology for LISP program construction from examples", Journal of the ACM 24, pp 161-175, 1977.

[8]富樫敦, 千葉和也, 野口正一, 「TRSプログラムの自動合成」 情報処理学会, ソフトウエア基礎論 31-1, 1989.