一种新的编译程序测试用例自动生成策略及其实现

吴 健 胡正国 蒋立源*

(西北工业大学计算机科学与工程系,西安 710072)

A NEW STRATEGY AND IT'S IMPLEMENTATION OF GENERATING AUTOMATICALLY SENTENCES FOR COMPILER TESTING

Wu Jian, Hu Zhengguo and Jiang Liyuan

(*Department of Computer Science and Engineering, Northwestern Polytechnical University, Xian* 710072)

Abstract Automatically generating testcase is one of approaches to test compilers. Nowadays Purdom strategy is normally used to do it. In this paper, we define a graph called DG graph to describe a context-free grammar, and based on the graph, we propose a new strategy to generate automatically sentences to test compilers. And also we describe the principles of the implementation of the new strategy.

摘要 自动生成测试用例是测试编译程序的途径之一. 目前一般均使用 Purdom 提出的产生式选择策略来自动生成测试用例. 本文提出了描述前后文无关文法的一种图表示方法,这种图称为文法的 DG 图. 基于 DG 图,我们给出了一种优于 Purdom 策略的新的编译程序测试用例自动生成策略,并讨论了这种策略的实现原理.

§0. 引 言

目前,编译程序测试用例的生成方式可以分为两类:人工编写和自动生成. 前者一般是组织一批有关专业人员,按照一定的原则,手工编制一组测试用例. 用这种方法编制的程序,针对性强,测试效率较高. 缺点是工作量大,且难以全面覆盖语言的语法成份. 后一类方式是构造一测试用例自动生成器,根据对象语言的文法,自动生成一组满足测试要求的测试用

* 本文 1991 年 3 月 11 日收到,1991 年 6 月 6 日定稿. 本课题受国家自然科学基金项目 6873039 和航空基金高等院校自选课题资助. 作者**吴健**,讲师,主要研究领域为软件工程. **胡正国**,教授,主要研究领域为软件工程. **蒋立源**,教授,主要研究领域为软件工程.

例.这一组测试用例能比较全面地覆盖源语言的语法成份,其缺点是针对性较差.本文主要讨论编译程序测试用例自动生成过程中的产生式选择策略问题.

迄今为止,已经出现了许多实验性的编译程序测试用例自动生成程序,例如[2—4]等.这些生成程序中,不管它们解决前后文相关性的方法有何不同,也不管其输入文法的表示方式有何差异,其绝大多数都是建立在 Purdom 算法基础上的.Purdom 算法本质上体现了这样一种策略,即要求在生成程序集合的过程中,文法的每一条产生式至少被使用一次,这是一种产生式覆盖策略.

通过具体的实践和理论分析我们发现,由于 Purdom 算法仅考虑了最简单的组合情况,所以,其测试能力会受到一定的限制.当然,产生式的各种组合情况一般来说是无穷的,但我们可以考虑一系列实现上可接受,然而比 Purdom 算法更加优越的生成策略.这就是本文所要讨论的问题.

§1. 前后文无关文法的图表示

下面,我们不妨假定所要讨论的都是已化简前后文无关文法(即其中不含无用产生式)[6],且在文法 $G[S]=(V_N,V_T,P,S)$ 中增加一条产生式:$P_0:S_0\to S$,而把 S_0 看成是新文法 G' 的开始符号(当然要求 $NOT(S_0\in V)$,这里 $V=V_N\cup V_T$).

定义 1:设有文法 $G=(V_N,V_T,P,S)$,对 V_N 中的每一元素 A,定义

$$FOLLOW(A)=\{B\mid S\overset{+}{\underset{LM}{\Rightarrow}}\beta A\gamma B\delta,\text{且 } B\in V_N,\gamma\in V_T^*,\beta,\delta\in V^*\}$$

这里$\underset{LM}{\Rightarrow}$表示最左推导.

文法非终结符的 FOLLOW 集可以用下面的规则进行迭代计算:

1)若 P 中有如下的产生式 $C\to\beta A\gamma B\delta$,其中,$A,B\in V_N,\gamma\in V_T^*,\beta,\delta\in V^*$,则 $B\in FOLLOW(A)$;

2)若 P 中有产生式 $C\to\alpha A\beta,A\in V_N,\beta\in V_T^*,\alpha\in V^*$,且 $B\in FOLLOW(C)$,则 $B\in FOLLOW(A)$.

为了后面的叙述清楚起见,我们把 P 中每一条产生式的左端符号显式地表示出来.

定义 2:设有文法 $G=(V_N,V_T,P,S)$,定义 $VP(G)=\{\langle A,f\rangle\mid f\in P,\text{且 A 是 f 的左端符号}\}$.显然有 $|VP(G)|=|P|$.

定义 3:设有文法 $G=(V_N,V_T,P,S)$,定义与 G 相关的图 DG(G)为(VP(G),E),称 VP(G)中的元素为图 DG(G)的结点,E 是该图的边集.且对任意的 $\langle A,f\rangle\in VP(G)$ 和 $\langle B,h\rangle\in VP(G)$,仅当下述条件之一成立,才有 $(\langle A,f\rangle,\langle B,h\rangle)\in E$:

1)f 形如 $A\to\alpha B\beta,\alpha\in V_T^*,\beta\in V^*$;

2)f 形如 $A\to\gamma,\gamma\in V_T^*$,且 $B\in FOLLOW(A)$.

例 1:设有一文法 $G=(\{S_0,S,E,T,P\},\{+,\uparrow,(,),i\},P,S_0)$,P 中的产生式如下:

$f_0:S_0\to S$　　$f_1:S\to(E)$　　$f_2:E\to E+T$　　$f_3:E\to T$

$f_4:T\to P\uparrow T$　　$f_5:T\to P$　　$f_6:P\to(E)$　　$f_7:P\to i$

则其 DG(G)如图 1 所示(容易求得 FOLLOW(P)={T}).

显然,如上定义的 DG 图反映了按最左推导方式生成句子的过程中,所使用产生式的序列.然而,容易看出,例 1 的图中没有出度为零的结点.因此,为了描述从图的指定结点 $\langle S_0,$

$f_0\rangle$出发，沿何种路径行进才能对应一个句子的最左推导，我们需要将图与一个控制字符串联系起来.

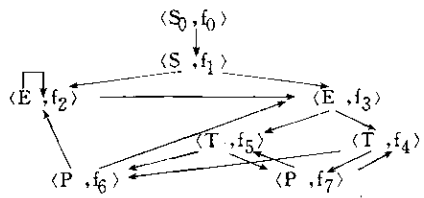

图1 一文法的DG图

定义 4：设有一文法 $G=(V_N,V_T,P,S)$，SENT 是 G 的一个句型. 对 DG(G)的任一结点$\langle A,f\rangle$：

1)若 $SENT=\alpha A\beta,\alpha\in V_T^*,\beta\in V^*$，则称结点$\langle A,f\rangle$可最左作用于句型 SENT(以下简称为作用). 若 f 形如 $A\rightarrow\gamma,\gamma\in V^*$，则称句型 $\alpha\gamma\beta$ 为其作用结果；

2)若 $SENT=\alpha\in V_T^*$；或 $SENT=\beta B\gamma,\beta\in V_T^*,\gamma\in_T^*,B\in V_N$，且 $B\neq A$，则称$\langle A,f\rangle$对 SENT 的作用无定义，或称$\langle A,f\rangle$不能作用于 SENT.

定义 5：设 $SENT_1=S_1$，S_1 为文法 G 的开始符号. 图 DG(G)中，从$\langle S_1,f_1\rangle\in VP(G)$和 $SENT_1$ 开始的一条路径是如下的一个句型－结点序列：$SENT_1,\langle S_1,f_1\rangle,SENT_2,\langle S_2,f_2\rangle,\cdots,SENT_n,\langle S_n,f_n\rangle,SENT_{n+1}$，其中：1)每一个$\langle S_i,f_i\rangle$可作用于 $SENT_i$，$1\leqslant i\leqslant n$；2)$SENT_{i+1}$是$\langle S_i,f_i\rangle$作用于 $SENT_i$ 的作用结果，$1\leqslant i\leqslant n$；3)$SENT_{n+1}\in V_T^*$.

我们称$\langle S_n,f_n\rangle$是该路径的终结结点，$SENT_{n+1}$是该路径的终结句型.

下面，我们不加证明地列出 DG 图的有关性质.

性质 1：设 $SENT_1,\langle S_1,f_1\rangle,SENT_2,\langle S_2,f_2\rangle,\cdots,\langle S_n,f_n\rangle,SENT_{n+1}$是 DG(G)图中从 $SENT_1,\langle S_1,f_1\rangle$开始的一条路径(这里 S_1 是 G 的开始符号，$SENT_1=S_1$)，则对任意 $1\leqslant i\leqslant n$，$\langle\langle S_i,f_i\rangle,\langle S_{i+1},f_{i+1}\rangle\rangle$是 DG(G)中的一条边.

此性质表明，DG(G)中的一条路径描述了一个句子的最左推导过程，也是有向图 DG(G)中在图论意义下的一条有向通路.

性质 2：设有文法 $G=(V_N,V_T,P,S)$，若 $\alpha\in L(G)$，则 DG(G)中存在一条从$\langle S,f_1\rangle$和 $SENT_1=S$ 开始的路径，该路径的终结句型是 α.

性质 3：DG(G)中的每一条边都是可历的(即存在一条路径经过该边).

§2. 基于 DG 图的产生式选择策略

根据 DG 图及其性质，我们可得到如下一些编译程序测试用例自动生成的策略.

策略 1(路径覆盖策略)：生成文法 G 的一句子集，使得 DG(G)中的每一条路径至少被通过一次.

由于 DG 图中通常存在回路，所以其中的路径数是无穷的，故这一策略是不现实的. 但从理论上来说，这是一条能力最强的策略.

策略 2(简单路径覆盖策略)：生成文法 G 的一句子集，使得 DG(G)中每一条简单路径至少被通过一次. 所谓简单路径是指路径中的回路仅需通过一次.

策略 3(边覆盖策略)：生成文法 G 的一句子集，使 DG(G)中每一条边至少被通过一次.

策略 4(结点覆盖策略)：生成文法 G 的一句子集，使得 DG(G)中每一个结点至少被通过一次. 本策略又可称为产生式覆盖策略.

Purdom 算法中的产生式选择策略即能满足策略 4 的要求.

设 ST 为一策略，PS 为程序集，若用 ST(PS)表示 PS 能满足 ST，则可以证明：

策略 1(PS)⇒策略 2(PS)⇒策略 3(PS)⇒策略 4(PS).

其中,"⇒"表示逻辑蕴涵,即若 PS 满足策略 i,则必满足策略 i+1,1≤i≤3.

§3. 边覆盖策略的实现

3.1 基本概念

为描述边覆盖策略的实现原理,我们首先给出一些基本概念和结论.

定义 6:设文法 $G=(V_N,V_T,P,S)$,对于 V_N 中的每一元素 A,定义集合 Last(A)如下:

1)对所有的 $A\in V_N$,$A\in Last(A)$;

2)若存在一产生式 $f\in P$,且 f 形如 $A\to\alpha B\beta$,$\alpha\in V^*$,$B\in V_N$,$\beta\in V_T^+$,则 $B\in Last(A)$;

3)若 $C\in Last(B)$,$B\in Last(A)$,则 $C\in Last(A)$.

定义 7:设有一文法 $G=(V_N,V_T,P,S)$,定义 V_N 上的二元关系 $Last^*$ 如:$Last^*=\{\langle A,B\rangle \mid A,B\in V_N$,且 $B\in Last(A)\}$.

显然,关系 $Last^*$ 从另一角度描述了 Last 集,它们本质上是等价的.由 Last 的定义知,$Last^*$ 是自反的且是可传递的,则可由下式来计算 $Last^*$:

1)$Last_0=\{\langle A,B\rangle \mid A,B\in V_N$,且 P 中存在产生式 $A\to\alpha B\beta$,$\alpha\in V^*$,$\beta\in V_T^+\}$;

2)$Last^*=\bigcup_{i=0}^{\infty}Last_0^i$.

此式可用 Warshell 算法[7]来计算.

定义 8:设有文法 $G=(V_N,V_T,P,S)$,定义 P 中每一元素的 pair 集如下:

1)对 $f\in P$ 且 f 形如 $C\to\alpha A\beta B\gamma$,$\alpha,\gamma\in V^*$,$\beta\in V_T^+$,$A,B\in V_N$,有$\langle A,B\rangle\in pair(f)$;

2)对 $f\in P$ 且 f 形如 $C\to\alpha D\beta$,$\alpha,\beta\in V^*$,且 P 中存在以 D 为左端的产生式 g,使$\langle A,B\rangle\in pair(g)$,则$\langle A,B\rangle\in pair(f)$;

3)若$\langle C,B\rangle\in pair(f)$,$f\in P$,且$\langle C,A\rangle\in Last^*$,则$\langle A,B\rangle\in pair(f)$;

4)此外,pair(f)中不含其它元素.

据此定义,可以构造一个求 pair 集的迭代算法,本文从略.

定理 1:设有文法 $G=(V_N,V_T,P,S)$,f 是 P 中的一条产生式,当且仅当存在一 V^* 中的串 $\alpha A\beta B\gamma$,$\alpha,\gamma\in V^*$,$\beta\in V_T^+$,且 $C(f)\overset{*}{\underset{LM}{\Rightarrow}}\alpha A\beta B\gamma$(C(f)表示 f 的左端符号为 C),有$\langle A,B\rangle\in pair(f)$.

3.2 边覆盖策略的实现算法

下面将在 pair 集及其性质的基础上,说明边覆盖策略的实现方法.首先,我们给出边覆盖算法的总的结构,然后再讨论其中的产生式选择问题.

边覆盖策略的句子生成算法

```
置 S0 于栈 STACK 中;{NOTS0∈V}
oldvertex:=⟨S0,S0→S⟩;{S 为文法的开始符号}
while 栈 STACK 不空 do
  begin
    弹出 STACK 的栈顶符号到 symbol 中;
    ifsymbol 是终结符
      then 输出 symbol
      else
```

```
        begin
          choose(symbol,P);
          {选择一条以 symbol 为左端的产生式 P}
          newvertex:=⟨symbol,P⟩;
          置边⟨oldvertex,newvertex⟩为已历过;
          oldvertex:=newvertex
        end
    end
```

本算法的核心问题是过程 choose(symbol,P)的设计. 算法中的 oldvertex 和 newvertex 表示 DG 图中的结点,而⟨oldvertex,newvertex⟩则表示 DG 图中的一条边.

下面,我们给出过程 choose(symbol,P).

CHOOSE(symbol,P)

```
{在以 symbol 为左端的产生式中,选择满足如下条件的产生式 P}
  if  边⟨oldvertex,⟨symbol,P⟩⟩尚未遍历过                                    (1)
  then 选择产生式 P
  else
  if  symbol(P)⇒αAβ,α,β∈V*,A∈VN,且存在一条以 A 为左端的产生式 Q,A(Q)→δBγ,δ∈Vt*,B∈VN,γ∈V*,且
      存在一条以 B 为左端的产生式 R,使边⟨⟨A,Q⟩,⟨B,R⟩⟩尚未遍历过,且 A 不在栈 STACK 中   (2)
  then 选择产生式 P
  else
      if  存在⟨A,B⟩∈pair(P),且存在一条以 A 为左端的产生式 Q:A→δ∈Vt*,及存在一条以 B 为左端的产生式 R:B
          →γ∈V*,而且边⟨⟨A,Q⟩,⟨B,R⟩⟩尚未遍历过                              (3)
      then 选择产生式 P
      else 选择能导致最短推导的产生式 P
```

考察文法的 DG 图,我们发现,图中的边可以分为两类:一类边⟨⟨A,f⟩,⟨B,g⟩⟩,其中产生式 f 形如 A→αBβ,α∈Vt*,β∈V*,我们将它们称为实边;另一类边⟨⟨A,f⟩,⟨B,g⟩⟩,其中产生式 f 形如 A→α∈Vt*,且 B∈FOLLOW(A),我们称其为虚边.

容易证明,过程 choose(symbol,P)中的条件(1)、(2)能确保遍历 DG 图中的实边;而根据定理 1,choose(symbol,P)中的条件(1)、(3)能确保遍历 DG 图中的虚边. 至于如何选择能导致最短推导的产生式,在[1]中已有阐述,本文不再给出.

结束语:我们已在 VAX8350 计算机上实现了一个编译程序测试用例自动生成器,该生成器以 Purdom 策略为依据选择产生式. 同时我们已将该生成器用于对 JOVIAL(J73)语言编译程序的测试,并成功地发现一些人工测试未能发现的错误. 目前我们正在已有工作的基础上,实现本文所给出的实现算法,并准备将本算法也用于对 JOVIAL 语言编译程序的测试,以期通过实践而不仅仅通过理论分析来证明:边覆盖策略确实优于目前通用的 Purdom 策略.

参考文献

1 Purdom P., A Sentence Generator for Testing Parsers, Bit 12, 1972.

2 W. Horner and R. Schovler, Independent Testing of Compiler Phases Using a Testcase Generator, Soft. Practice and Experience, Jan. 1989.

3 F. Bazzich and I. Spadofora, An Automatic Generator for Compiler Testing, IEEE Trans. on Soft. Eng., July 1982.

4 Celenlano A., Compiler Testing Using a Sentence Generator, Soft. Practice and Experience, Nov. 1980.

5 P. A. Luker, Program Generator and Generator Software, The Computer Journal, 4, 1988.

6 John E. Hopcroft and Jeffrey D. Ullman, Introduction to Automata Theory, Languages and Computation, Addison — Wesley Publishing Company, 1979.

7 D. J. Cooke, H. E. Bez, Computer Mathmatics, Combridge University Press, 1984.