基于线性阵列网络的并行语法分析算法设计

孙玉强[1,2] 孙玉珊[2] 刘三阳[1] 张英丽[2]

(西安电子科技大学理学院 西安710071)[1] (江苏工业学院计算机系 常州213000)[2]

摘 要 本文提出了线性阵列LA(Linear Array)连接状态中上下文无关文法(CFG)的并行语法分析算法的设计思想,指出对形如$[i,j,B\rightarrow\eta\cdot]$的项目传递时环形拓扑结构的冗余,并以实例详细描述了线性阵列连接结构中分析存储信息的演变过程。

关键词 上下文无关文法,并行算法,语法分析

Algorithm Design of Parallel Parsing Based on Linear Array

SUN Yu-Qiang[1,2] SUN Yu-Shan[2] LIU San-Yang[1] ZHANG Ying-Li[2]

(School of Science, Xidian University, Xi'an, 710071)[1]

(Department of Computer, Jiangsu Polytechnic University, Changzhou, 213000)[2]

Abstract The design thoughts of parallel parsing context-free grammar connecting in linear Array are proposed in this paper, and it points out that the useless of the structure in circle when passing the items of the form $[i,j,B\rightarrow\eta\cdot]$, and it portrays in detail the transferring courses of parsing storing information in linear-array structure with examples.

Keywords Context-free grammar, Parallel algorithm, Syntax parsing

1 引言

随着时代的进步和科技的发展,在各高新领域对高性能并行计算的研究与开发越来越受到人们的高度重视,已成为世界各国科技竞争的战略制高点。并行机能否有效发挥并行计算能力,其并行编译器的性能占举足轻重的地位。为超高性能并行计算机生成高速运行的并行代码,是并行编译程序的主要任务。并行编译技术及优化从理论到实践都有许多课题需要进行深入地研究,已成为当前计算机学科并行处理技术中最重要、最活跃的研究领域之一。

并行语法分析是并行优化编译技术中的主要环节和重要组成部分,主要用于并行程序设计语言的编译器和解释器的实现,以及自然语言的理解和翻译等领域中[1~3]。在文[4]中,按传统的语法分析规则,描述了对一般上下文无关语言的并行识别和语法分析的并行算法。文[5]对上下文无关语言的特殊集括号语言,给出了一个用$n/\log n$个处理器和$O(\log n)$时间的识别和语法分析优化的并行算法。对于任意的上下文无关文法在线性环形拓扑结构中的并行语法分析算法,在文[6]中有较深入的讨论,并且给出了在多处理机环境下并行语法分析的项目传递过程中单向环形拓扑结构的优势和分析算法。本文提出了线性阵列LA(Linear Array)连接状态上下文无关文法的并行语法分析算法的设计思想,指出对形如$[i,j,B\rightarrow\eta\cdot]$的项目传递时环行拓扑结构的冗余,并以实例详细描述了线性阵列连接结构中存储信息的演变过程。

2 相关定义

定义1 线性阵列LA(Linear Array)又称为一维连接,其连接方式是并行机中最基本的互连方式,其中每个处理机只与其左,右近邻相连,所以也叫做二近邻连接,这种连接方式是心动结构的最基本形式。N个节点用$N-1$条边串接,内节点度为2,直径为$N-1$,对剖宽度为1,当首尾处理机相连时,可构成循环移位连接,在拓扑结构上等同于环。

约定处理机的数目为n个,P为处理机的地址,则线性阵列的连接函数LC可定义为:

$LC_{-1}(P)=P-1 \quad 1\leqslant P\leqslant n-1$

$LC_{+1}(P)=P+1 \quad 0\leqslant P\leqslant n-2$

其中,$LC_{-1}(P)$和$LC_{+1}(P)$分别表示左和右连接。

例1 对于文法$G:E\rightarrow E+E|E*E|(-E)|-E|i$,输入的字符串

$i*(-i+i)$用8个处理机PE_i,$i=0,1,2,\cdots,7$,按线性阵列的网络结构表示为图1:

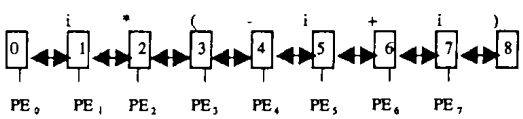

图1

定义2 上下文无关文法(CFG)定义为一个五元组$G=\{V_N,V_T,\Sigma,P,S\}$其中:

V_N表示非终结符集合,其元素用大写字母$A,B,C,\cdots$(可以带下标,以下同)表示;

V_T表示终结符集合,其元素用小写字母$a,b,c,\cdots$表示;

Σ表示所有符号的集合,即$\Sigma=V_N\cup V_T$,$|V_N|$表示CFG文法中的非终结符的个数,$|V_T|$表示CFG文法中的终结符的个数,$V_N\cap V_T=\phi$,V_T^*表示终结符连接组成的串的集合(包

*)河南省基础研究(004061800)、自然科学基金(0211021600和0324220079)资助项目。孙玉强 教授,博士生,主要研究方向为软件工程、并行计算与优化。孙玉珊 副教授。刘三阳 教授,博导。

括空串 ϕ)，集合 $\Sigma^*=(V_N\cup V_T)^*$ 表示由非终结符和终结符连接组成的串的集合，其元素用希腊字母 $\alpha,\beta,\gamma,\cdots$ 表示；

P 表示产生式的集合，其中每个产生式是形如 $A\to\alpha$ 的推导规则，这里 $A\in V_N,\alpha\in\Sigma^*$，其中 A 称为产生式的左部，α 称为产生式的右部，且不允许出现形如 $A\to A$ 的产生式；

S 表示文法的开始符号，且 $S\in V_N$。

本文提到的文法，如未加说明都是指上下文无关文法(CFG)。

定义3 我们用 $\alpha\Rightarrow\beta$ 表示：存在 γ,δ,η 和 A，满足 $\alpha=\gamma A\delta,\beta=\gamma\eta\delta$，且 $A\to\eta$ 是 P 中的一个产生式。

对于串 α,β，如果存在 $\alpha_1,\alpha_2,\cdots,\alpha_m\in\Sigma^*(m\geqslant 1)$，使得 $\alpha\Rightarrow\alpha_1\Rightarrow\alpha_2\Rightarrow\cdots\Rightarrow\alpha_m\Rightarrow\beta$，则称 $\alpha\Rightarrow^*\beta$，并且称序列 $\alpha_1,\alpha_2,\cdots,\alpha_m$ 为从 α 到 β 的一个推导。

如果存在推导过程 $S\Rightarrow^*\beta$，则称 β 为 G 的一个句型。如果 $\beta\in V_T^*$，则称 β 为文法 G 的句子。文法 G 的所有句子构成的集合称为文法 G 的语言 $L(G)$，即 $L(G)=\{\alpha\mid\alpha\in V_T^*,S\Rightarrow^*\alpha\}$，即语言 $L(G)$ 是 V_T 上的一个串集，空字用 ϕ 表示，空字满足：

$\phi u=u\phi=u$，并且 $|\phi|=0$

对于一个输入字符串 $\omega=b_1b_2\cdots b_n,b_i\in V_T^*(i=1,2,\cdots,n)$ 可以用图表示该串，在每相邻的两个符号 b_i 和 b_{i+1} 之间用顶点 i 来表示。顶点0在 b_1 的前面，顶点 n 在 b_n 的后面。

定义4 对于一个给定的上下文无关文法 G 和一个输入字符串 $\omega,\omega=a_1a_2\cdots a_n$，其中 $a_i\in V_T^*(i=1,2,\cdots,n)$，如果存在 P 中的产生式 $A\to\alpha\beta$，并且 $\alpha\Rightarrow^* a_{i+1}a_{i+2}\cdots a_j$，那么定义三元组 $I=[i,j,A\to\alpha\cdot\beta]$ 为相应于 ω 的一个项目(item)，简称为项目。项目 I 所跨越的终结符的个数定义为项目 I 的长度，记作 $|I|$，即 项目 $I=[i,j,A\to\alpha\cdot\beta]$ 的长度记为 $|I|=|j-i|$。对应于 ω 的任意项目 $[i,j,A\to\alpha\cdot\beta]$ 都满足 $0\leqslant i\leqslant j\leqslant n-1$。项目 $[i,j,A\to\alpha\cdot\beta]$ 可以用图表示，表示的方法为：画一条从顶点 i 开始，指向顶点 j 的带权弧线，弧线上标出项目 $A\to\alpha\cdot\beta$，如图2所示。

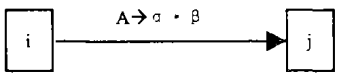

图2

例2 对于文法 $G:E\to E+E|E*E|-E|(E)|i$。$N=\{E\},\Sigma=\{+,*,-,(,),i\}$，$S=E,P=\{E\to E+E,E\to E*E,E\to -E,E\to(E),E\to i\}$，输入终结字符串 $\omega=i*(-i+i)=a_1a_2\cdots a_8$。由文法 G 的产生式：$E\to i$，可得 $E\Rightarrow i=a_1$，可得项目 $[0,1,E\to i\cdot]$；由文法 G 的产生式：$E\to E+E$ 和 $E\to i$，可得 $E\Rightarrow i=a_1$，可得项目 $[0,1,E\to E\cdot+E]$；由文法 G 的产生式：$E\to E+E$ 和 $E\to i$，可得 $E\Rightarrow^* i+i=a_5a_6a_7$，可得项目 $[4,7,E\to E\cdot+E]$ 等等。

3 $IL(i)$结构描述与定理

对于一个给定的上下文无关文法 G，和一个长度为 n 的输入字符串 $\omega=b_1b_2\cdots b_n$(其中 $b_i\in V_T^*,i=1,2,\cdots,n$)，采用并行语法分析的方法，分别存储这些项目，开始时，把从顶点 i 出发的项目，分别存放在标记为：$PE_0,PE_1,PE_2,\cdots,PE_{n-1}$ 的 n 个处理机中。

定义5 把所有的存储在处理机 PE_i 中的项目组成的集合表示为 $IL(i)$。由项目(item)的定义，很容易得出项目有如下的性质：

假设一个给定的上下文无关文法 G，输入的字符串长度为 n，对于 $0\leqslant i,j,k\leqslant n,A,B\in V_N,\alpha,\beta,\eta\in\Sigma^*,a\in V_T$，有下列定理成立：

定理1 如果 $[i,k,A\to\alpha\cdot B\beta]\in IL(i),[k,j,B\to\eta\cdot]\in IL(j)$，那么 $[i,j,A\to\alpha B\cdot\beta]\in IL(i)$。

证明：由 $[i,k,A\to\alpha\cdot B\beta]\in IL(i)$，得 $A\to\alpha B\beta\in P$，

并且 $\alpha\Rightarrow^* a_{i+1}a_{i+2}\cdots a_k$，(1)

又由 $[k,j,B\to\eta\cdot]\in IL(j)$，得 $B\to\eta\in P$，

并且 $\eta\Rightarrow^* a_{k+1}a_{k+2}\cdots a_j$ (2)

由(1)，(2)可得：$\alpha B\Rightarrow^* a_{i+1}a_{i+2}\cdots a_kB\Rightarrow a_{i+1}a_{i+2}\cdots a_k\eta\Rightarrow a_{i+1}a_{i+2}\cdots a_k\,a_{k+1}\,a_{k+2}\cdots a_j$，故 $[i,j,A\to\alpha B\cdot\beta]\in IL(i)$ □

定理2 如果 $[i,j,B\to\eta\cdot]\in IL(i)$，且 $A\to B\beta\in P$，那么 $[i,j,A\to B\cdot\beta]\in IL(i)$。

证明：由 $[i,j,B\to\eta\cdot]\in IL(i)$，得 $B\to\eta\in P$，并且 $\eta\Rightarrow^* a_{i+1}a_{i+2}\cdots a_j$。故 $B\Rightarrow\eta\Rightarrow^* a_{i+1}a_{i+2}\cdots a_j$

又由 $A\to B\beta\in P$，故 $[i,j,A\to B\cdot\beta]\in IL(i)$ □

定理3 如果 $[i,j-1,A\to\alpha\cdot a\beta]$，且 $a=a_j$，那么 $[i,j,A\to\alpha_a\cdot\beta]\in IL(i)$。

证明：由 $[i,j-1,A\to\alpha\cdot a\beta]$，可得 $A\to\alpha a\beta\in P$，并且 $\alpha\Rightarrow^* a_{i+1}a_{i+2}\cdots a_{j-1}$，又由 $a=a_j$，故 $\alpha a\Rightarrow\alpha a_j\Rightarrow^* a_{i+1}a_{i+2}\cdots a_{j-1}a_j$，故 $[i,j,A\to\alpha a\cdot\beta]\in IL(i)$ □

4 LA 上并行语法分析算法设计

4.1 算法描述与说明

由项目的定义易得，对所有的 $S\to a\beta\in P$，如果 $a\in V_T$，并且 $a=b_{i+1}$，那么将 $[i,i+1,S\to a\cdot\beta]$ 加入到 $IL(i)$ 中，由项目的性质1可得，如果 $[i,k,A\to\alpha\cdot B\beta]\in IL(i),[k,j,B\to\eta\cdot]\in IL(j)$，那么 $[i,j,A\to\alpha B\cdot\beta]\in IL(i)$。但是 $[k,j,B\to\eta\cdot]\in IL(k)$ 存储在处理机 PE_k 的 $IL(k)$ 中，所以在下一步开始之前，把 $[k,j,B\to\eta\cdot]$ 从 PE_k 转移到 PE_i，即每台处理机从它的后继处理机接受一些归约项目，在同步等待所有的处理机完成上一步骤后，进入下一步骤。重复地执行转移和加入过程，直到没有新的元素可以加入到 $IL(i)$ 中为止。

详细的并行算法可参看文[1]，但是经过观察，不难看出文[1]中的并行语法分析算法有多余的计算。

定理4 该算法在第 k 步($1\leqslant k\leqslant n$)，构造长度为 k 的项目，当由待约项目扩充处理机 PE_i 的 $IL_{(i)}$ 中的元素时，处理机 $PE_i(0\leqslant i\leqslant n-k-1)$ 接受从其后继处理机 $PE_{succ(i)}$ 的 $IL_{(succ(i))}$ 中转移过来的归约项目，如果 $k+i\geqslant n$，那么处理机 $PE_i(k+i\geqslant n)$ 没有必要接受后继处理机 $PE_{succ(i)}$ 的 $IL_{(succ(i))}$ 中的归约项目。

证明：因为在第 k 步($1\leqslant k\leqslant n$)，处理机 PE_i 的 $IL_{(i)}$ 中的元素，即每一个项目 $[i_1,i_2,A\to a\cdot\beta]$ 的长度 $\leqslant k$，而项目 $[i_1,i_2,A\to a\cdot\beta]$ 的起点 $i_1\geqslant i$，所以有 $k+i_1\geqslant k+i\geqslant n$，即扩充后的项目，其终点超过所输入的字符串的长度 n，即溢出。

4.2 算法设计

定义函数 $E^*=(G,n,b_1\cdots b_n)$

输入：上下文无关文法 G；正整数 n；字符串 $b_1b_2\cdots b_n$。

输出：函数 E，如果 $b_1b_2\cdots b_n$ 是 $L(G)$ 某个长度为 n 的句子，那么返回真；否则返回假。

```
begin
(1)for i=0 to n-1 par-do
   IL(i)=φ;
     for all A→a β∈P par-do
       if a∈V_T and a= b_{i+1}
         add [i, i+1,A→a·β] into IL (i);
       end if
     end for
   end for
(2)for i=0 to n-1 par-do
       If[i,i+1, B→η·]∈IL(i) and A→B β∈P par-do
       add [i,i+1,A→B·β] into IL(i)
```

```
      end if
    end for
(3)for i=0 to n-1 par-do
     for k=2 to n par-do
      If i+k≤n then par-do
      (3.1) for r=1 to k-1 par-do
              if [i,i+r,A→α·Bβ]and[i+r , i+k , B→η·]∈IL(i)
                then add [i,i+k,A→αB·β]into IL(i)
             end if
            end for
      (3.2)if[i,i+k,B→η·]∈IL(i) then
             for each production A→Bβ∈P par-do
               add [i,i+k,A→B·β] into IL(i)
             end for
           end if
      (3.3)if [i,i+k-1,A→α·aβ]∈IL(i), and a=a_{i+k} then add
                        [i,i+k,A→αa·β] into IL(i)
           end if
     end if
    end for
    end for
(4)for all IL(i) par-do
       for all I∈IL(i)
           if |I|=n then return yes
           end if
       end for
    end for
    return no
end
```

4.3 构造过程分析

我们举例说明并行构造 $IL(i)$结构的项目集的过程：

例3 对于文法 $G:E\to E+E|E*E|-E|(E)|i$，输入的终结字符串为 $\omega=i*(-i+i)=a_1a_2\cdots a_8$，由并行构造 $IL(i)$结构的项目集的步骤：

(1)初始状态 $k=1$，构造 $IL(i)$结构中长度为1的项目集；

(2)并行传送；

(3)并行添加。

按照构造 $IL(i)$结构项目集的步骤，向 $IL(i)$结构项目集添加项目，构造 $IL(i)$结构的详细过程由图3和表1显示。由算法，程序运行的过程图可以表示为：

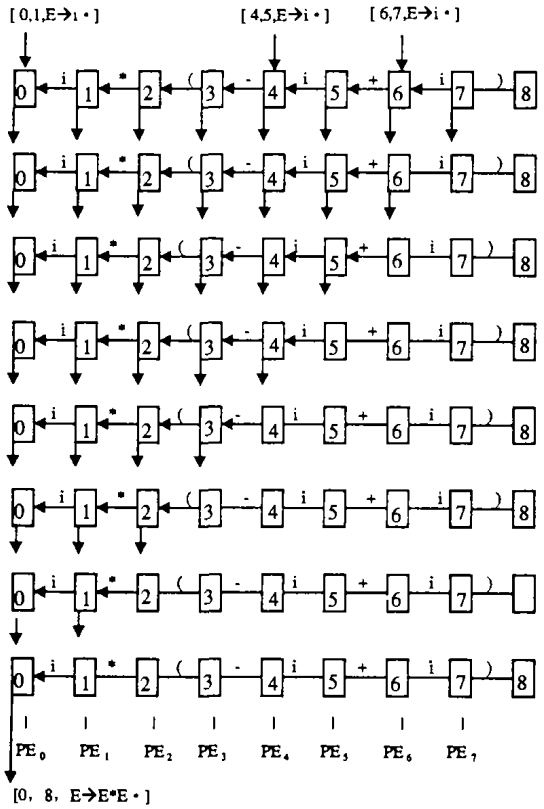

图3

图3中，水平向左的箭头表示在第 k 步处理机 PE_i ($0\leqslant i\leqslant n-k-1$)接受从其后继处理机 $PE_{succ(i)}$的 $IL_{(succ(i))}$)中可能转移过来的归约项目的传递途径；垂直向下的箭头表示第 k 步 $IL(i)$现行状态经过第 $k+1$步处理后处理机 PE_i 由归约、移进和待约项目按照定理2扩充而得的新的移进项目和待约项目及规约项目(在 $IL(i)$中)而达到的新的现行状态的传递。

表1 $IL(i)$中的部分变化

	$IL(0)$	$IL(1)$	$IL(2)$	$IL(3)$
并行添加 $k=4$				[3,7, $E\to E\cdot+E$] [3,7,$E\to E\cdot*E$] [3,7,$E\to E+E\cdot$]
并行传送	[3,5,$E\to -E\cdot$] [4,5,$E\to i\cdot$]		[6,7,$E\to i\cdot$] [4,7,$E\to E+E\cdot$] [3,7,$E\to E+E\cdot$]	
并行添加 $k=5$			[2,7,$E\to (E\cdot)$]	
并行传送		[6,7,$E\to i\cdot$] [4,7,$E\to E+E\cdot$] [3,7,$E\to E+E\cdot$]		

表1显示了该过程中从第4步到第5步 $IL(i)(i=0,1,2,3)$中新的添加内容。重复地并行执行转移和加入过程，经过8个步骤后，最后运行的结果为[0,8,$E\to E*E\cdot$]$\in IL(0)$，这就说明：输入的字符串 $i*(-i+i)\in L(G)$。

结束语 本算法的设计比较带环形拓扑结构的方法虽然占用的处理机数目仍为 n，即等于所输入的字符串数目，但是在第 $k(1\leqslant k\leqslant n)$步，由于仅考虑前 $n-k+1$台处理机中的有效项目，结构相对简单并且没有增加复杂度，其相应数据结构及转换函数相对简洁。算法的进一步优化是后续研究希望解决的问题。

参 考 文 献

1 Ibarra O H,Pong T-C,Sohn S M. Parallel recognition and parsing on the hypercube. IEEE Trans. Comput,1991,40:764～770

2 EarleyJ. An efficient context-free parsing algorithm. Communications of the ACM,1970,13(2):94～102

3 Sakakibara Y. Learning context-free grammars from structural data in polynomial time. Theoretical Computer Science,1990,76:223～242

4 Gibbons A,Rytter W. Efficient parallel algorithms. Cambridge University Press,1990

5 Rytter W,Giancarlo R. Optimal parallel parsing of bracket language. Theoretical Computer Science,1987

6 Ra Dong-Yu,Kim Jong-Hyu,A parallel parsing algorithm for arbitrary contexe-free grammars ,Information Processing Letters,1996,58:87～96