国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
一九八七年第二期　总第五十八期　No.2 1987 Sum. 58

逻辑程序并行执行模型及其体系结构的研究

（梗　概）

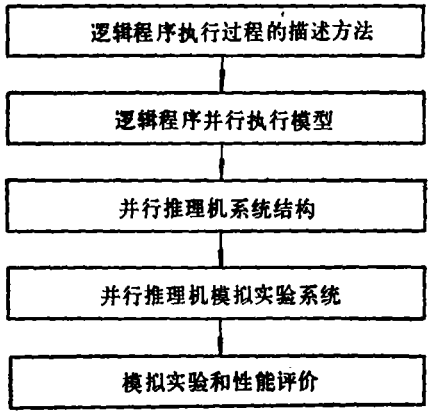
论文作者：孙成政
指导教师：慈云桂　教授
（计算机系）

一、引　言

1.1 研究目标、方式和结果

目标：并行推理机系统结构。

方式：自顶向下。



结果：1. OR 树林描述方法；
2. PSOF 并行执行模型；
3. PIM-PSOF 并行推理机系统结构；
4. SES-PIM 模拟实验系统。

本文1987年3月收到

1.2 逻辑程序并行处理的基本概念

1.2.1 逻辑程序设计与 Horn 子句

1.2.2 逻辑程序的不确定性与并行性

不确定性的两种意义：

(1) 当一个子目标可与多个子句的头部匹配时，选择这些子句的策略是不确定的；

(2) 当一个子句体中存在多个子目标时，执行这些子目标的方式是不确定的。

两种基本并行性：(1) OR 并行性；

(2) AND 并行性。

一个重要条件：共享变量的约束值必须一致。可能的 AND 并行执行方式：

(1) 无条件并行执行方式：低效，不实用；

(2) 流并行执行方式：通讯开销严重；

(3) 不相关子目标并行执行方式：有前途的方式。

1.2.3 逻辑程序的执行过程与描述方法

执行过程：从某初始目标出发，使用某些推理规则，推出新目标的过程。

描述方法：1. OR 树（SLD 搜索树）方法；

2. AND/OR 树方法。

1.3 逻辑程序并行处理研究领域的代表性工作

1. 并发逻辑程序设计语言的研究；

2. 逻辑程序并行执行模型及其系统结构的研究。

1.3.1 Pollard 的并行执行模型

1981 年，英国帝国理工学院 Pollard 博士。

核心概念："调解" (Reconciliation)机制。

主要贡献：第一个并行执行 Horn 子句的模型，开创性的工作。

存在问题：1. 调解方法低效；

2. 没有涉及系统结构方面的问题；

3. 没有做任何实验工作。

1.3.2 AND/OR 进程模型

1983 年，美国加州大学(Irvine)，Conery 博士。基本框架与 Pollard 的模型相同。

主要贡献：1. 第一次提出了用数据相关性分析方法有选择地开发 AND 并行性的思想和算法；

2. 用 DEC-10 PROLOG 实现了一个基于 AND/OR 模型的解释器，第一次用模拟实验的方法证明了并行解释逻辑程序的可行性。

主要问题：1. 排序算法开销很大；

2. 对存储管理等问题未做研究。

1.3.3 RAP 模型

RAP(Restricted AND-Parallelism)

1984 年，美国 IBM 公司 Watson 研究中心 DeGroot 博士。

基本思想：用编译程序对逻辑程序进行预处理。

主要贡献：第一次提出了用静态分析和动态测试相结合的方法开发逻辑程序中AND并行性。

存在问题：1. AND 并行性的开发程度受限；

2. 没对开发 OR 并行性提供支持。

1.3.4 TOKEN 模型

1984 年，瑞典皇家工学院，Ciepielewski 博士。

基本思想：通过对 OR 树的并行搜索实现对逻辑程序的并行执行。

主要贡献：第一个基于 OR 树的并行执行模型。

存在问题：1. 不能支持 AND 并行执行；

2. OR 并行执行中存在大量冗余。

1.3.5 G-R 模型

G-R(Goal-Rewriting)

1984 年，日本，东京大学，Goto 博士。

基本框架：与 TOKEN 模型相同。

主要特点：1. 结构复制和环境复制；

2. 并行推理机(PIE)的具体结构；

3. 大量软/硬件模拟实验。

1.3.6 其它模型

- 数据流(Dataflow)模型；
- 归约(Reduction)模型。

二、OR 树林描述方法

2.1 传统 OR 树方法分析

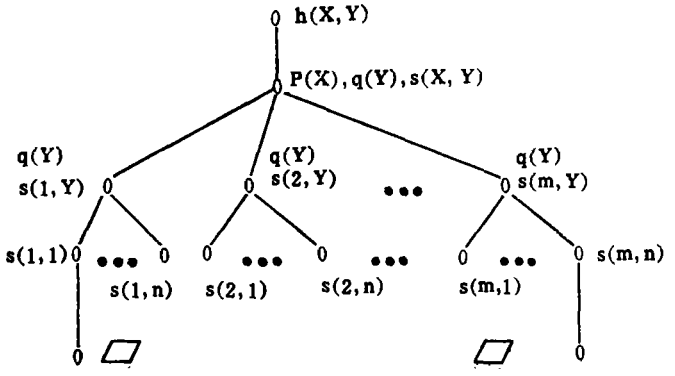
?-h(X,Y)

h(X,Y)：-p(X), q(Y), s(X,Y).

p(1). p(2). … p(m).

q(1). q(2). … q(n).

s(1,1). s(m,n).



主要特点：

1. 每一条分枝代表一条独立的求解路径；

2. 不能描述 AND 并行执行；

3. OR 并行执行中存在大量冗余。

2.2 OR 树林方法

2.2.1 OR 树林方法的几个定义

定义：串行一步推理规则(SOIR)。

给定目标 $G=A1, A2,\cdots,Ai-1, Ai, Ai+1,\cdots,Am$

$$m>1,\ 1\leqslant i\leqslant m$$

如果 1. Ai 是从G中选出的子目标；

2. 程序中存在子句：

$$P0:\ -P1,\cdots,Pn\quad n\geqslant 0$$

满足：$P0\theta=Ai\theta$

其中 θ 为 P0 与 Ai 的最一般一致化代换。

则可由本规则推出新目标

$$G'=(A1,\cdots,Ai-1,\ P1,\cdots,Pn,\ Ai+1,\cdots,Am)\theta。$$

定义：并行多步推理规则(PMIR)。

给定目标G，

如果 1. G含有 n 组相互独立的子目标：

$$g1,\ g2,\cdots,gn\quad n>1;$$

2. θi 是 gi 的一个解，$1\leqslant i\leqslant n$；

则可按本推理规则推出新目标：

$$G'=(G-(g1\cup g2\cup\cdots\cup gn))\theta$$

这里 $\theta=\theta1\cup\theta2\cdots\cup\theta n$。

定义：OR 树林。

给定一个逻辑程序，一个初始目标，一个划分规则和一个选择规则，这个程序的执行过程可以唯一地描述成一个 OR 树林。这个 OR 树林由若干 OR 树组成。树中的节点用目标标记并按如下方式产生：

1. 初始目标用于标记一棵树的根节点；

2. 对任一用目标G标记的节点，如果G是一个空子句目标，则这个节点是一个成功的叶节点。否则，用给定的划分规则将G中的子目标划分成如下各组：

$$g0,\ g1,\ g2,\cdots,gn\quad n\geqslant 0$$

其中，g1，g2，…，gn 相互独立，g0 与任一 gi(i>0)相关。

(1) 如果 n=0，用给定的选择规则从G中选出一个子目标A，从程序中找出可与A匹配的全部子句，Ci，$1\leqslant i\leqslant k$。

a. 如果 k=0，G 标记的节点是一个失败的叶节点。

b. 如果 k>0，G 标记的节点有 k 个后继节点，标记这些后继节点的目标 G'i($1\leqslant i\leqslant k$)是由 G 与 Ci 按 SOIR 推出的新目标。

(2) 如果 n>1，G 标记的节点成为一个种子节点并由此派生 n 棵树，每棵树的根分别由 g1，g2，…，gn 标记。假定由 gi(1=≤i=≤n)标记的树含有 Ki 个成功的叶节点，令 K=K1×K2×…×Kn。

a. 如果 K=0，G 标记的节点成为一个失败的叶节点。

b. 如果 K>0，G 标记的节点有 K 个后继节点。标记这些后继节点的目标 Gi(1≤i≤K)是由 G 按 PMIR 推出的新目标。

2.2.2 举例

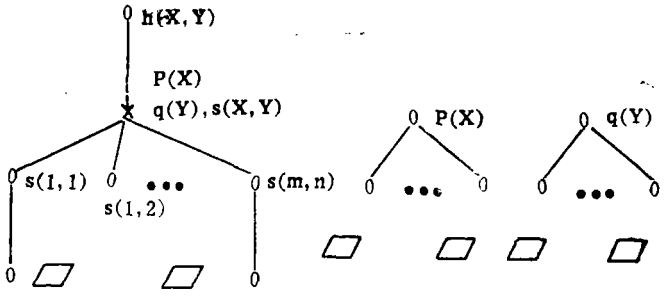
?-h(X, Y)

h(X, Y): -p(X), q(Y), s(X, Y)

p(1), p(2),…,p(m).

q(1), q(2),…,q(n).

s(1, 1), s(m, n).



主要特点：

1. 各条分枝之间的关系描述了 OR 并行执行；

2. 多棵 OR 树之间的关系描述了 AND 并行执行；

3. 一组独立子目标的执行用一棵分离的 OR 树描述，避免了传统 OR 树方法中的冗余计算。

三、PSOF 并行执行模型

3.1 OR 树林搜索模型的分类

Branch	Tree	Parallelism	Class
S	S	Non	SS-Model
P	S	OR	PS-Model
S	P	AND	SP-Model
P	P	AND-OR	PP-Model

S：顺序搜索方式　　P：并行搜索方式

3.2 子目标的自动划分

3.2.1 设计原则

P1. 满足I/O模式的要求;

P2. 遵守独立子目标并行执行策略;

P3. 尽可能多地避免冗余计算。

3.2.2 划分条件

C1. 假定子目标Ai和Aj共享变量且i<j，经划分后，如果在同一组或任一个在g0组，那么有：ORDER(Ai)←ORDER(Aj)。

C2. 对gi中任一子目标A，gj中任一子目标B，1≤i<j≤n，A和B必须相互独立。

C3. 对gi中任一子目标A，1≤i≤n，如果gi中有多个子目标，那么A必与gi中另一子目标共享变量。

C4. 对g0中任一子目标A，A必须与任一gi(1≤i≤n)中至少一个子目标相关，或者与g0中另一子目标B相关且满足ORDER(B)←ORDER(A)。

3.2.3 划分算法

定义：partition(IL, TL, OL)

```
If IL=[ ], then OL=TL and EXIT.
  Otherwise call first(IL, First),
            call rest(IL, Rest).
    If TL contains group(0, L0), then
      If First is independent of any group in TL, then
        call combine(TL, Combinedgroup),
        call number(NewT),
        Newgroup=group(NewT, [First]),
        NewTL=[combinedgroup, Newgroup],
        call partition(Rest, NewTL, OL).
      Otherwise if First is dependent of only group(T, Lt), then
        call entergroup(T, First, TL, NewTL),
        call partition(Rest, NewTL, OL).
      Otherwise if First is dependent of group(0, L0)or every group but
      group(0, L0)in TL, then
        call entergroup(0, First, TL, NewTL),
        call partition(Rest, NewTL, OL).
      Otherwise First must be dependent of k groups g1, g2,…,gk, k is
      greater than 1 and less than the length of TL, then
        call delete([g1,…,gk], TL, TL1),
        call number(NewT),
```

call combine([g1,…,gk, group(NewT, [First])], Combinedgroup),
call append(TL1, [Combinedgroup], NewTL),
call partition(Rest, NewTL, OL).

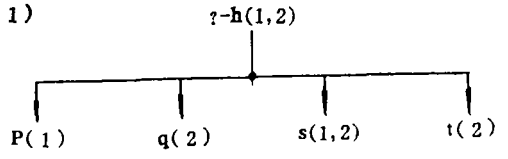
Otherwise if First is independent of any group in TL, then
call number(NewT),
Newgroup=group(NewT, [First]),
call append(TL, [Newgroup], NewTL),
call partition(Rest, NewTL, OL).

Otherwise if First is dependent of only group(T, Lt), then
call entergroup(T, First, TL, NewTL),
call partition(Rest, NewTL, OL).

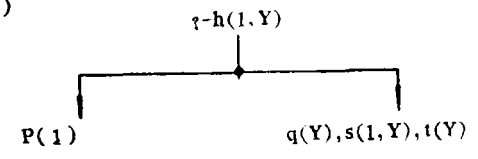
Otherwise if First is dependent of every group in TL, then
Newgroup=group(0, [First]),
call append(TL, [Newgroup], NewTL),
call partition(Rest, NewTL, OL).

Otherwise First must be dependent of k groups g1, g2,…,gk, k is

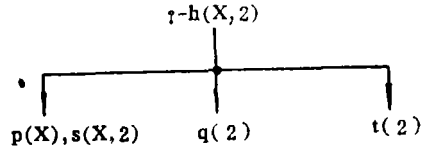
(1)



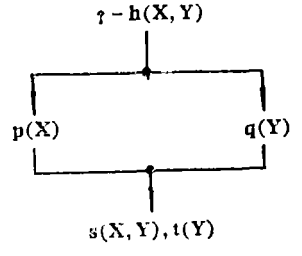
(2)



(3)



(4)



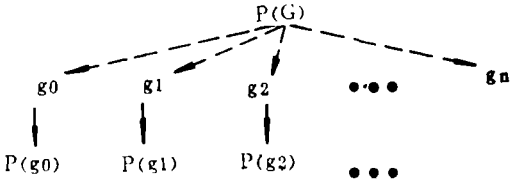
greater than 1 and less than the length of TL, then
call delete([g1,…,gk], TL, TL1),
call number(NewT),
call combine([g1,…,gk, group(NewT, [First])], Combinedgroup),
call append(TL1, [Combinedgroup], NewTL),
call partition(Rest, NewTL, OL).

3.2.4 举例

h(X, Y): -p(X), q(Y), s(X, Y), t(Y) (续见第20页之图)

3.3 派生树的并行搜索与 AND 并行执行

3.3.1 子进程的派生



3.3.2 子进程的控制

一) 数据结构

1. 进程计数器(PC);
2. 成功计数器(SC);
3. 解向量(SV)。

二) PC, SC 和 SV 的管理

1. 初始化: PC=1, SC=0, SV=[ ]。
2. m 条分枝的节点: PC+(m-1)⟹PC。
3. 叶节点: PC-1⟹PC。
4. 成功叶节点: SC+1⟹SC。

三) 控制条件

1. PC=0: 搜索完毕;
2. SC=0: 无解;
3. (PC=0)∧(SC=0): 搜索完毕且无解。

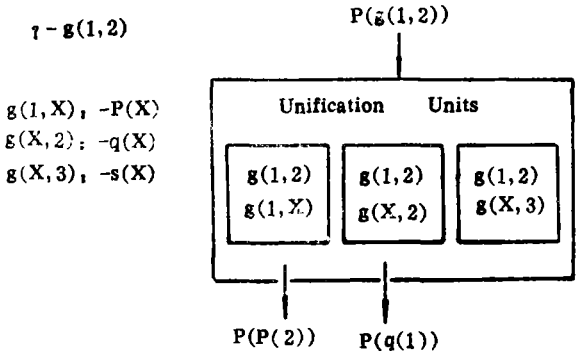
3.3.3 解的合成条件与算法

条件: SC1×SC2×…×SCn>0。

算法:
```
FOR I1=1 TO SC1 DO
  FOR I2=1 TO SC2 DO
    ⋮
    FOR Ii-1=1 TO SCi-1 DO
      FOR Ii+1=1 TO SCi+1 DO
        ⋮
```

FOR In=1 TO SCn DO
COMBINE(SV1(I1), SV2(I2),…,
SVi-1(Ii-1), SVi(SCi),
SVi+1(Ii+1),…,SVn(In))。

3.4 分枝的并行搜索与 OR 并行执行



3.5 否定目标 not 的并行执行

- 否定推理规则(NIR);
- 扩充的 OR 树林,
- 对扩充的 OR 树林的并行搜索。

3.6 “砍树”机制的嵌入

- 必要性
- “砍树”条件: PC=0∧SC=0(AND)
  SC=1(NOT)
- “砍树”规则

定义: 进程命名规则 1。

1. 初始进程命名为(n1);
2. 对名为(nj, nj-1,…,n1)的进程, 后继进程命名为(nj, nj-1,…,n1);
3. 对名为(nj, nj-1,…,n1)的进程, 子进程命名为 (nj+1, nj, nj-1,…,n1).

定义: 进程删除规则。

当名为(nj, nj-1,…,n1)的进程到达某叶节点时, 只要“砍树”条件成立, 则可以发出命令删除名字以(nj, nj-1,…,n1)为后缀的进程。

3.7 “剪枝”机制的嵌入

- 必要性
- “剪枝”条件: CUT, GUARD
- “剪枝”规则

定义: 进程命名规则 2。

1. 初始进程命名为(n1)。
2. 对名为(nj, nj-1,…,n1)的进程, 后继进程按如下三种情况命名:

(1) 如果执行的目标是 Guard，则命名为(nj-1,…,n1)；

(2) 如果匹配的子句中含 Guard，则命名为(nj+1，nj，nj-1,…,n1)；

(3) 其它情况，命名为(nj，nj-1,…,n1)。

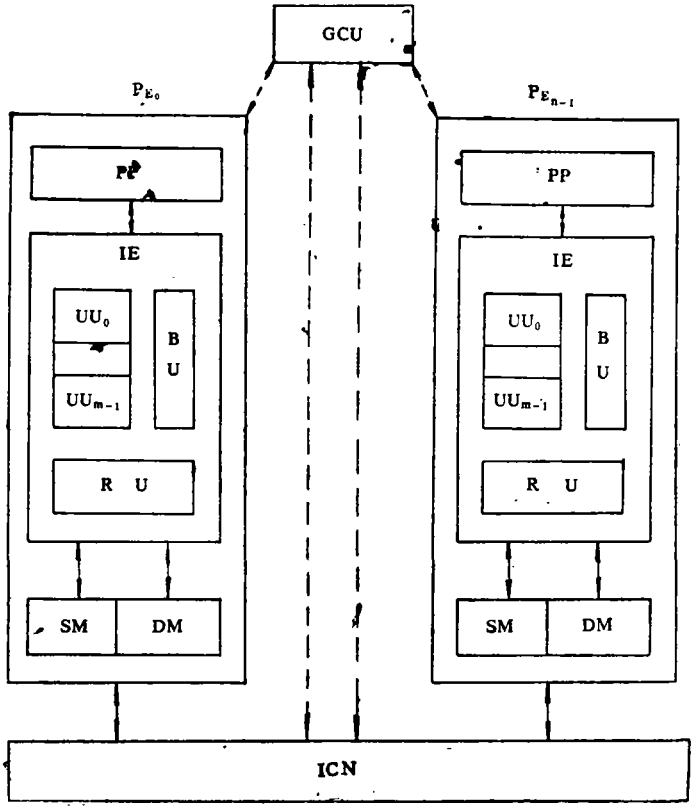
3. 对名为(nj，nj-1,…,n1)的进程，子进程命名为(nj+1，nj，nj-1,…,n1)。

定义：进程撤消规则。

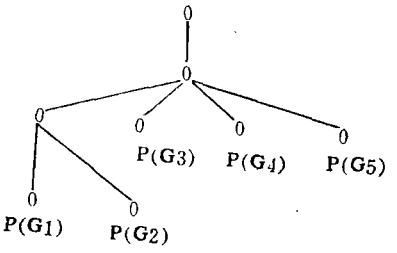
当名为(nj，nj-1,…,n1)的进程执行 Guard 目标时，则可发出命令撤消名字以(nj，nj-1,…,n1)为后缀的进程。

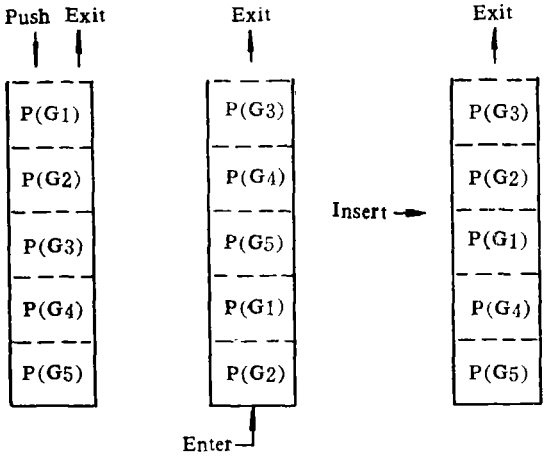
四、PIM-PSOF 并行推理机

4.1 系统结构



4.2 进程调度





4.3 任务分配

1. 请求分配策略；
2. 集中控制策略；
3. 最小状态选择策略；
4. 直接传送策略。

4.4 进程控制块

1. 进程名指针(PN)； 2. 文字目录指针(LD)； 3. 环境目录指针(ED)；
4. 种子节点控制块指针(SCB)； 5. 根节点控制指针(RCB)。

4.5 静态文字的表示

LDS	PS
LD = (EN1, EN2, ···, ENn)	CR1: H1: -P1, ···, Pk1
EN = (T, ENN, CR, LR)	CR2: H2: -Q1, ···, Qk1
T = 0/1	⋮
ENN: a unique integer	CRn: Hn: -S1, ···, Skn
CR: Clause Reference	
LR { a positive integer when T = 0 / a list of positive integer when T = 1	

特点：1. 不同子目标可共享相同的静态文字；

2. 不同的进程可共享静态文字。

4.6 动态环境的管理

EDS	BRS
ED = (EN1, EN2, …, ENn)	BR: (BV1, BV2, …, BVm)
EN = (ENN, LRR)	BV: (VAR, TERM, ENNT)
ENN: a unique integer	
LRR = (RR1, RR2, …, RRk)	
RR = (T, R)	
T: 1/0	
R: reference to BR in BRS	

不同进程可共享相同指针指向的约束记录。

环境管理过程：

1. BOUND (ED, VAR, ENNV, TERM, ENNT)
2. BINDING (ED, VAR, ENNV, TERM, ENNT)
3. UNIFY (ED, TERM1, ENN1, TERM2, ENN2)
4. COMPACTDIR (ED, LVAR, ENN, NED)
5. MERGEDIR (EDm, LED, NED)

4.7 控制信息的表示

1. 根节点控制块

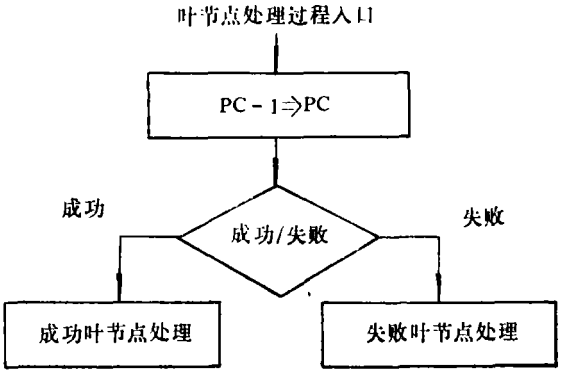
(1) 进程计数器(PC)； (2) 成功计数器(SC)； (3) 解向量(SV)。

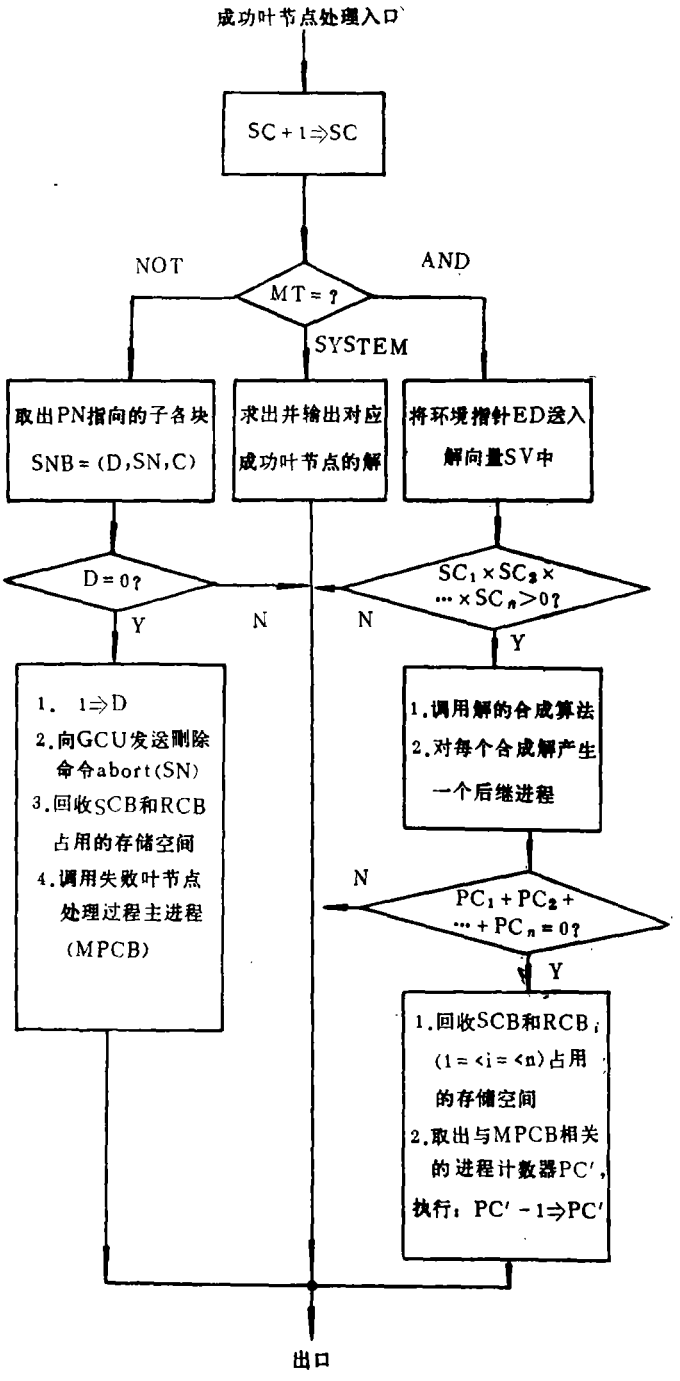
2. 种子节点控制块

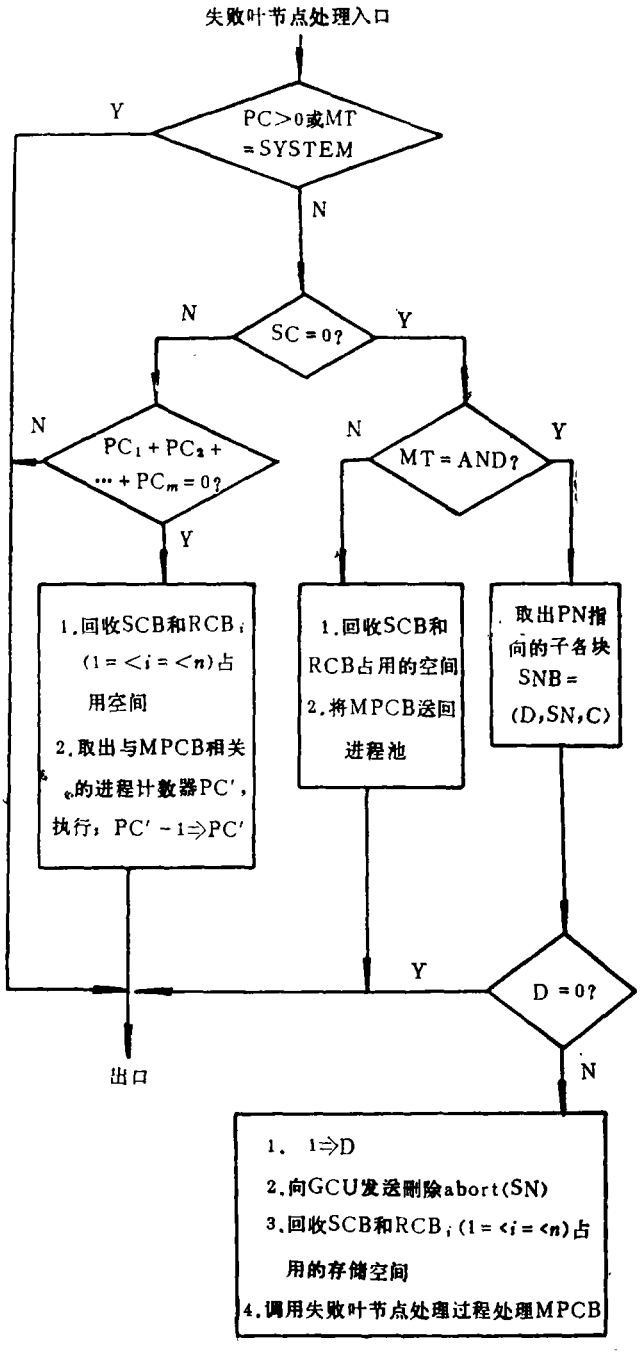
(1) 主进程类型(MT)； (2) 主进程控制块(MPCB)；

(3) 根节点指针表(RCBL)。

4.8 叶节点处理过程




成功叶节点处理入口
SC+1⇒SC
NOT
MT=?
AND
SYSTEM
取出PN指向的子各块
SNB=(D,SN,C)
求出并输出对应
成功叶节点的解
将环境指针ED送入
解向量SV中
D=0?
Y
N
$SC_1 \times SC_2 \times \cdots \times SC_n > 0$?
N
Y
1. 1⇒D
2.向GCU发送删除
命令abort(SN)
3.回收SCB和RCB
占用的存储空间
4.调用失败叶节点
处理过程主进程
(MPCB)
1.调用解的合成算法
2.对每个合成解产生
一个后继进程
N
$PC_1 + PC_2 + \cdots + PC_n = 0$?
Y
1.回收SCB和RCB,
(1=<i=<n)占用
的存储空间
2.取出与MPCB相关
的进程计数器PC',
执行: PC'-1⇒PC'
出口


失败叶节点处理入口
PC>0或MT=SYSTEM
Y
N
SC=0?
N
Y
$PC_1+PC_2+\cdots+PC_m=0$?
N
Y
MT=AND?
N
Y
1.回收SCB和RCB$_i$ ($1=<i=<n$)占用空间
2.取出与MPCB相关的进程计数器PC',执行; PC'-1⇒PC'
1.回收SCB和RCB占用的空间
2.将MPCB送回进程池
取出PN指向的子名块SNB=(D,SN,C)
D=0?
Y
N
出口
1. 1⇒D
2.向GCU发送删除abort(SN)
3.回收SCB和RCB$_i$ ($1=<i=<n$)占用的存储空间
4.调用失败叶节点处理过程处理MPCB

五、SES-PIM 模拟实验系统

5.1 研制目的

1. 验证算法的正确性和可行性；
2. 提供研究并行推理机的实验工具。

5.2 SES-PIM 系统主要特点

1. 客观性； 2. 系统性； 3. 灵活性； 4. 方便性。

5.3 五个分系统简介

1. SES-PIM-PSOF 分系统； 2. SES-PSOF 分系统；
3. SES-PIM-PSOT 分系统； 4. SES-PSOT 分系统；
5. SES-SIM-SSOT 分系统。

5.4 P-PROLOG 语言的简介

1. 语法上与 G-PROLOG 相同；
2. 子句的书写次序与执行次序无关；
3. 子目标执行次序由子目标划分机构自动确定（对用户透明）；
4. 包含大部分 G-PROLOG 系统的内部谓词。

5.5 系统参数

1. 处理单元(PE)个数：N。
2. 一台 PE 中一致化部件个数：M。
3. 初始目标个数：K。
4. 求解个数：S。
5. 搜索策略：SS。

六、实验结果与分析

计划的实验专题：

1. PSOF，PSOT，SSOT 模型的对比实验。
2. PIM-PSOF，PIM-PSOT 推理机的对比实验。
3. 一致化部件数与候选子句数以及系统性能之间的关系。
4. 内部谓词执行频度的统计分析。
5. 多道逻辑程序运行及其任务级并行性开发实验。
6. 各种搜索策略对系统性能的影响。
7. 各种任务分配算法对系统性能的影响。
8. 处理单元之间通讯模式和频度的统计分析及其对互连网络性能的要求。
9. 环境压缩算法的效果及其开销分析。
10. 环境共享和文字共享方案的性能分析。
11. 动/静态数据相关性分析算法的对比及其 AND 并行性开发的开销问题。

表 6.1 PSOF, PSOT, SSOT 三种模型的对比实验

问 题	模 型	搜索步数	平均并行度	速度比		并行度比	
				PSOT	SSOT	PSOT	SSOT
皇后	PSOF	19	19.21	2.21	13	3.26	19.21
	PSOT	42	5.88		5.88		5.88
	SSOT	247	1				
迷宫	PSOF	17	10.59	2.06	6.65	3.28	10.59
	PSOT	35	3.23		3.23		3.23
	SSOT	113	1				
矩阵乘	PSOF	30	3.23	3.17	3.2	3.2	3.23
	PSOT	95	1.01		1		1.01
	SSOT	96	1				
矩阵乘	PSOF	13	6.08	6.08	6.08	6.08	6.08
	PSOT	79	1		1		1
	SSOT	79	1				
快速求阶乘	PSOF	14	6.14	4	4	5.21	6.14
	PSOT	56	1.18		1		1.18
	SSOT	56	1				
快速分类	PSOF	33	3.82	2.67	2.97	3.14	3.82
	PSOT	88	1.22		1.11		1.22
	SSOT	98	1				

表 6.2 PIM-PSOF 推理机与 PIM-PSOT 推理机的对比实验（略）

表 6.3 候选子句数目的统计分析

问题类型	候选子句个数	出现次数	出现频度	候选子句平均个数
皇后	1	1	0.01	2.43
	2	66	0.55	
	3	52	0.44	
迷宫	1	1	0.01	3.25
	2	76	0.64	
	3	14	0.12	
	7	27	0.23	
矩阵乘	2	52	1.00	2.00
快速阶乘	1	1	0.05	1.95
	2	19	0.95	
快速分类	2	40	0.58	2.42
	3	29	0.42	

表6.4 内部谓词执行频度的统计分析

问题类型	内部谓词执行次数	子目标执行总数	内部谓词执行频度
快速阶乘	66	105	0.63
皇后	216	563	0.38
矩阵乘	27	131	0.21
快速分类	38	262	0.15
迷宫	0	700	0.00

表 6.5 多道逻辑序运行与任务级并行质开发实验

运行方式	执行步数	PE平均使用率	系统平均并行度	系统波动度
两道程序并行求解	37	0.66	5.24	1.60
快速分类单独求解	34	0.46	3.71	1.79
快速阶乘单独求解	16	0.58	4.63	2.70

七、结论与未来的工作

7.1 结论

1. 提出了一种描述逻辑程序执行的新方法——OR 树林方法。为建立能高效开发 AND 和 OR 两种并行性的执行模型提供了新的框架。

2. 提出了一种 OR树林并行搜索模型——PSOF模型；研究了子目标的自动划分，搜索进程的产生和控制等重要专题；嵌入了“砍树”和“剪枝”等控制机制，提供了支持 not，guard 等特殊目标的执行机制。

3. 设计了基于 PSOF 模型的并行推理机——PIM-PSOF；研究了 PIM-PSOF 的系统结构，工作原理，进程表示，进程调度，任务分配以及存储管理等专题，并设计了相应的文字共享和环境共享方案。

4. 用 PROLOG 语言，对有关算法进行了严格的描述。以此为基础，在 VAX-11/780 上设计并实现了一个并行推理机模拟实验系统——SES-PIM。

5. 借助于 SES-PIM 系统，对用多处理机实现并行推理的若干专题进行了综合的实验研究，为今后并行推理机样机的设计提供了有价值的实验数据。

7.2 未来的工作

1. 利用编译技术高效地实现子目标自动划分算法；
2. 进一步扩充 PSOF 模型以支持各种特殊目标的执行；
3. 研究对 OR 树林的各种启发式搜索算法；
4. 在 SES-PIM系统上实现多种并发逻辑程序设计语言。

参 考 文 献

[1] Sun Chengzheng and Tzu Yungui, The OR-forest Description for the Execution of Logic Programs, Lecture Notes in Computer Science, the Proc. of the Third International Conference on Logic Programming (July, 1986), pp. 710-717.

[2] Sun Chengzheng and Tzu Yungui, A New Method for Describing the AND-OR-parallel Execution of Logic Programs, To Be Published by Journal of Computer Science and Technology.

[3] Sun Chengzheng and Tzu Yungui, PSOF: A Process Model Based on the OR-forest Description, Proc. of the International Conference on Computer and

Communication (October, 1986).PP.457—466.

[4] Tzu Yungui and Sun Chengzheng, Study of Computation Model and Computer Architecture for Parallel Execution of Logic Programs, Invited Lecture in the International Confenrece on Computer and Communication (October, 1986).

[5] Sun Chengzheng and Tzu Yungui, PIM-PSOF: A Parallel Inference Machine Based on PSOF Model, the Second National Conference on Logic Programming, 1986, China.

[6] Sun Chengzheng and Tzu Yungui, SES-PIM: A Simulation and Experiment System for PIM-PSOF, the Second National Conference on Logic Programming, 1986, China.

[7] Sun Chengzheng, The EPM2I Interconnection Functions and the SUS Network, Proc. of the First International Conference on Computers and Applications(June, 1984), pp. 571-576.

[8] Sun Chengzheng and Tzu Yungui, An Automatic Partition Algorithm for AND-parallel Execution in the Framework of OR-forest, To appear in the Proc. of the Second International Conference on Computers and Applications (June, 1987).

[9] Sun Chengzheng and Tzu Yungui, The Sharing of Environment in AND-OR-parallel Execution of Logic Programs, Proc. of the 14th International Symposium on Comouter Architecture (June,1987).

[10] Wang Ping, Sun Chengzheng and Tzu Yungui, Parallel Execution of Negative Goal in the Extended PSOF Model, Proc. of the Second International Conference on Computers and Applications (June,1987).

[11] Tzu Yungui and Sun Chengzheng, Parallel Execution of Lcgio Programs in the Framework of OR-forest, Invited Paper in the 1987 Fall Joint Conference on Computer (Dallas, Octorber, 1987).

Study of Computation Model and Computer Architecture for Parallel Execution of Logic Programs

Sun Chengzheng

Abstract

The goal of our research is to design a computer architecture for AND-OR-parallel execution of logic programs based on the Horn clause subset of the predicate logic. We start from abstract level and move step by step towards concrete implementation details.

The major contributions made in this thesis are:

1. OR-forest—a new method for describing the execution of logic programs.
2. PSOF—a process model based on parallel search of OR-forest.
3. PIM-PSOF—an abstract parallel inference machine based on PSOF model.
4. SES-PIM—a simulation and experiment system for parallel inference machine.