華中科技大學

人工智能课程设计报告

选题名称： 产生式系统求解汉诺塔问题

实验时段：2019 年 11 月-2019 年 12 月

指导教师：冯 琪

专业班级：CS1601

学　　号：U201614526

姓　　名：田志伟

计算机科学与技术学院

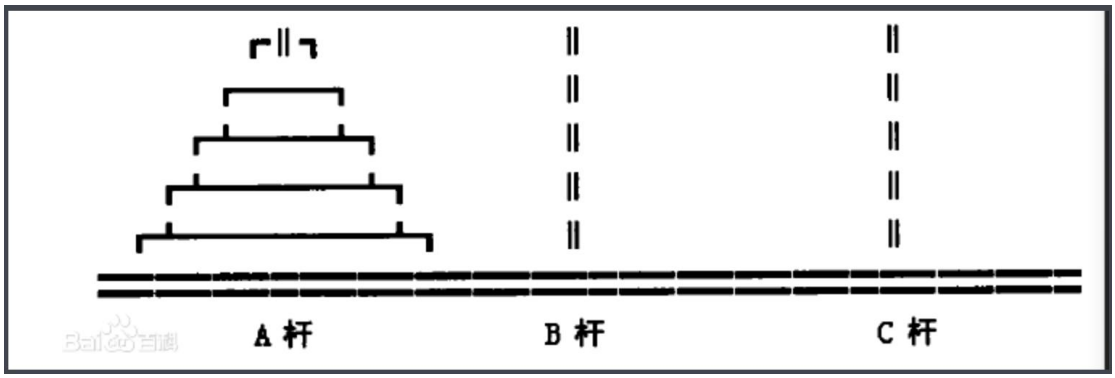
目录

一、实验目的

掌握人工智能产生式系统的基本思想，应用这种思想解决汉诺塔问题，并与传统递归方式求解进行对比。

二、问题描述

在一块铜板装置上，有三根杆(编号 A、B、C)，在 A 杆自下而上、由大到小按顺序放置 n 个金盘(如下图)。游戏的目标：把 A 杆上的金盘全部移到 C 杆上，并仍保持原有顺序叠好。操作规则：每次只能移动一个盘子，并且在移动过程中三根杆上都始终保持大盘在下，小盘在上，操作过程中盘子可以置于 A、B、C 任一杆上。



三、算法实现

1.递归算法：可以将汉诺塔问题分解为将 n-1 个盘子转移到 B 杆，最后 1 个盘子转移到 C 杆的子问题，调换 A，B 杆的位置递归求解。

源码：

```
func (han *hanota) moveDish(level, from, inter, to int) {
o1++
if level == 1 {
    han.s[to-1].Push(han.s[from-1].Pop().Value)
    fmt.Printf("Move dish %d from %d to %d.\n", level, from, to)
} else {
    han.moveDish(level-1, from, to, inter)
    han.s[to-1].Push(han.s[from-1].Pop().Value)
    fmt.Printf("Move dish %d from %d to %d.\n", level, from, to)
    han.moveDish(level-1, inter, from, to)
}
```

```
}
```

2.产生式系统：

1）产生式系统：产生式系统是指认知心理学程序表征系统的一种。为解决某一问题或完成某一作业而按一定层次联结组成的认知规则系统。由全局数据库、产生式规则和控制系统三部分组成。每一产生式规则由条件(即当前的状态或情境)和行动两部分组成,其基本规则是“若条件 X，则实施行动 Y”，即当一个产生式中的条件得到满足，则执行该产生式规定的某个行动。

2）对汉诺塔问题制定如下产生式规则：

A.在移动盘子时，每次只移动可移动的最大的盘子

B.如果一个盘子上一次已经移动过，则不可以再移动

C.将盘子移动前的状态入栈，如果无盘子可移或每种移动方式均失败则返回 false，回溯状态栈

D.当要移动的最大目标盘子为 1 的时候，直接将其移至 C，返回 true，算法结束

3）数据结构

```
type hanota struct {
    n, pre, done int              //n 汉诺塔初始盘子数,pre 前一次移动的盘子的位置,初始
                                    值为 0,done 记录汉诺塔已经成功移动的盘子数
    s            [3]*stack.Stack //用栈来存储汉诺塔每个柱子的情况
}
func newHanoi(n int) *hanota {} //创建一个汉诺塔盘
func (han *hanota) dishToMove() (int, []int) { //求出汉诺塔中下一个需要移动的盘子
                                                编号,以及移动的目标个数
func (han *hanota) resolve(hanoiStack *stack.Stack) bool {} //产生式系统求解
```

四、实验结果分析

对于四层汉诺塔问题的求解过程如下，

递归算法：

```
[ladlod@mbp~/H/c/h/a/src> go run main.go
4
Move dish 1 from 1 to 2.
Move dish 2 from 1 to 3.
Move dish 1 from 2 to 3.
Move dish 3 from 1 to 2.
Move dish 1 from 3 to 1.
Move dish 2 from 3 to 2.
Move dish 1 from 1 to 2.
Move dish 4 from 1 to 3.
Move dish 1 from 2 to 3.
Move dish 2 from 2 to 1.
Move dish 1 from 3 to 1.
Move dish 3 from 2 to 3.
Move dish 1 from 1 to 2.
Move dish 2 from 1 to 3.
Move dish 1 from 2 to 3.
0 0 1
0 0 2
0 0 3
0 0 4
```

产生式算法:

```
Move dish 1 from 1 to 2.
Move dish 2 from 1 to 3.
Move dish 1 from 2 to 1.
Move dish 2 from 3 to 2.
Move dish 1 from 1 to 2.
Move dish 3 from 1 to 3.
Move dish 1 from 2 to 1.
Move dish 2 from 2 to 3.
Move dish 1 from 1 to 2.
Move dish 2 from 3 to 1.
Move dish 1 from 2 to 1.
Move dish 3 from 3 to 2.
Move dish 1 from 1 to 2.
Move dish 2 from 1 to 3.
Move dish 1 from 2 to 1.
Move dish 2 from 3 to 2.
Move dish 1 from 1 to 2.
Move dish 4 from 1 to 3.
Move dish 1 from 2 to 1.
Move dish 2 from 2 to 3.
Move dish 1 from 1 to 2.
Move dish 2 from 3 to 1.
Move dish 1 from 2 to 1.
Move dish 3 from 2 to 3.
Move dish 1 from 1 to 2.
Move dish 2 from 1 to 3.
Move dish 1 from 2 to 3.
true
0 0 1
0 0 2
0 0 3
0 0 4
```

并且可以看出，递归算法共进行了 15 次移动，而产生式算法共进行了 27 次移动，由于在移动盘子过程中可能会选择错误路径，在时间复杂度上递归算法要优于产生式算法。

五、实验总结

本次实验是对我们本学习人工智能这门课学习成果的检验，让我们在实践中熟悉了人工智能的算法。

在实验开始阶段，我的产生式并没有设计移动最大可移动的盘子，而是从 A->C 遍历，出现了死循环的现象，让我认识到了对于产生式系统，如果产生式设计不够完善，系统是无法准确完成对工作任务的判断的，我们在人工智能的设计中也应当做到对所有情况考虑完善，才能更好地让人工智能为我们服务。

六、源码

见 https://github.com/ladlod/hust_cs_ug/tree/master/aiExp