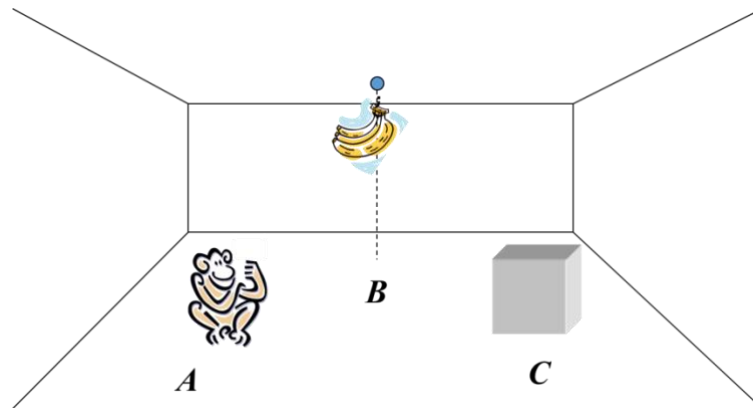


一、问题描述

1.1 待解决问题的解释

一个房间里，天花板上挂有一串香蕉，有一只猴子可在房间里任意活动（到处走动，推移箱子，攀登箱子等）。设房间里还有一只可被猴子移动的箱子，且猴子登上箱子时才能摘到香蕉，问猴子在某一状态下（设猴子位置为 A，箱子位置为 B，香蕉位置在 C），如何行动可摘取到香蕉。



1.2 问题的形式化描述

定义一个综合数据库 (M, B, Box, On, H) 存储不同状态，其中：

M: 猴子的位置

B: 香蕉的位置

Box: 箱子的位置

On=0: 猴子在地板上

On=1: 猴子在箱子上

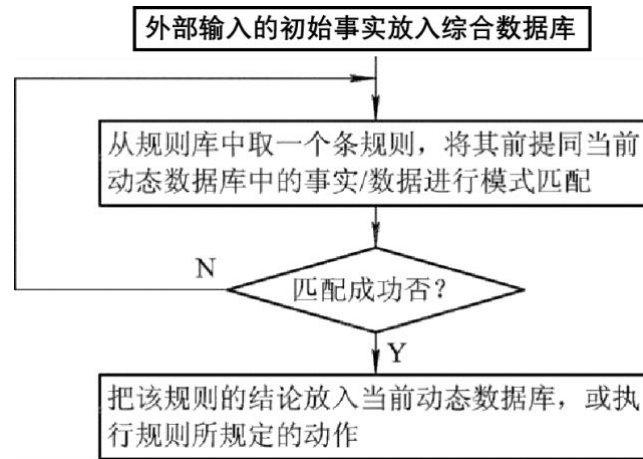
H=0: 猴子没有抓到香蕉

H=1: 猴子抓到了香蕉

故问题转化为寻找从初始状态 $(a, c, b, 0, 0)$ 到结束状态 $(c, c, c, 1, 1)$ 的路径。

1.3 解决方案原理

产生式系统的运行过程如下：



利用产生式系统，定义规则集：

r1: IF (x, y, z, 0, 0) THEN (w, y, z, 0, 0) (猴子移动)

r2: IF (x, y, x, 0, 0) THEN (z, y, z, 0, 0) (猴子推箱子)

r3: IF (c, c, c, 0, 0) THEN (c, c, c, 1, 0) (猴子爬箱子)

r4: IF (c, c, c, 1, 0) THEN (c, c, c, 1, 1) (猴子抓香蕉)

其中， x, y, z 为变量。

随后根据具体问题将规则具体化，找到得到香蕉的路径：

IF (a, c, b, 0, 0) THEN (b, c, b, 0, 0)

IF (b, c, b, 0, 0) THEN (c, c, c, 0, 0)

IF (b, c, b, 0, 0) THEN (b, c, b, 1, 0)

IF (c, c, c, 1, 0) THEN (c, c, c, 1, 1)

二、算法介绍

2.1 所用方法的一般介绍

在代码实现过程中，参照原理，对每个状态定义略有改动：

Status 定义： (loc_monkey, loc_box, loc_banana, on_box, hold, father)

loc_monkey: 猴子的位置

loc_box: 箱子的位置

loc_banana: 香蕉的位置

on_box: 猴子在地板上(0), 在箱子上(1)

hold: 没有抓到香蕉(0), 抓到了香蕉(1)

father: 指向上一个状态

初始状态为 (a, b, c, 0, 0, -1) , 终止状态为(c, c, c, 1, 1, item_father)。

规则集由两个操作来反映: goto(self, index) 和 push_box(self, index)。

2.2 算法伪代码

使用广度优先搜索算法, 在由初始状态展开的状态空间中搜索目标状态:

(1)把初始状态放入线性表;

(2)检查该线性表是否为空, 若为空, 则问题无解, 失败退出;

(3)把线性表的第一个节点取出, 标记为 n, 并标记为已经拓展过的节点;

(4)考察节点 n 是否为目标状态, 若是, 则得到问题的解成功退出;

(5)若节点 n 不可扩展, 则转第(2)步;

(6)通过 goto 和 push_box 操作扩展节点 n, 将其子节点放入线性表的尾部, 并为每个子节点设置指向父节点的指针, 转向第(2)步。

三、算法实现

3.1 实验环境和问题规模

实验环境: Python 3.7(64 bit), PyCharm 2019.3.3

Python interpreter:

Python 3.7 E:\APP\anaconda\python.exe

问题规模:

①时间复杂度为 $O(1)$, 因为问题步骤显而易见, 能在常数步内得到解;

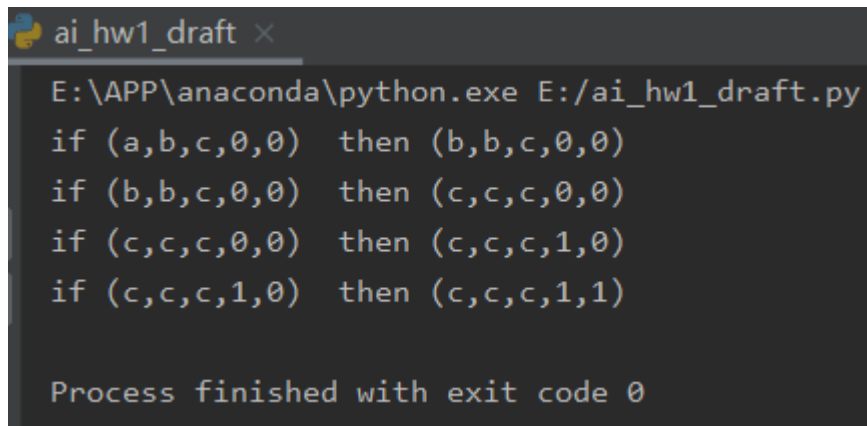
②空间复杂度为 $O(1)$ 。

3.2 数据结构

Status 结构体定义: (loc_monkey, loc_box, loc_banana, on_box, hold, father) ,

其中 father 类似于线性表的指针。

3.3 实验结果



```
ai_hw1_draft x
E:\APP\anaconda\python.exe E:/ai_hw1_draft.py
if (a,b,c,0,0) then (b,b,c,0,0)
if (b,b,c,0,0) then (c,c,c,0,0)
if (c,c,c,0,0) then (c,c,c,1,0)
if (c,c,c,1,0) then (c,c,c,1,1)

Process finished with exit code 0
```

3.4 系统中间及最终输出结果

同 3.3，即为问题的解决过程。

四、总结讨论

这次的实验主要在于知识的表示，可以用产生式来表示知识，并且设定情景里的状态和规则集之后，就可以通过搜索来解决这一问题。虽然这个猴子摘香蕉问题比较简单，但是以后遇到复杂的问题，也可以采用类似的方法解决问题。

参考文献

[1]王万森.人工智能原理及其应用[M]. 北京：电子工业出版社，2012.09: 32-33.

附录：源代码及其注释

```
import numpy as np
```

```
site = ['a', 'b', 'c']
```

```
class status:
```

```

def __init__(self, loc_monkey='a', loc_box='b', loc_banana='c',
on_box=0, hold=0, father=-1):

    self.loc_monkey = loc_monkey

    self.loc_box = loc_box

    self.loc_banana = loc_banana

    self.on_box = on_box

    self.hold = hold

    self.father = father


def goto(self, index):

    new_status = []

    if self.on_box == 1:

        return new_status

    for item in site:

        if item != self.loc_monkey:

            new_status.append(status(loc_monkey=item,
loc_box=self.loc_box, on_box=self.on_box, hold=self.hold, father=index))

    return new_status


def push_box(self, index):

    new_status = []

    if self.on_box == 1 or self.loc_box != self.loc_monkey:

        return new_status

    for item in site:

        if item != self.loc_monkey:

            new_status.append(status(loc_monkey=item, loc_box=item,
on_box=self.on_box, hold=self.hold, father=index))

    return new_status


def is_equal(status1, status2):

```

```

    if __name__ == '__main__':

        if status1.loc_monkey==status2.loc_monkey            and
status1.loc_box==status2.loc_box                            and
status1.loc_banana==status2.loc_banana                    and
status1.on_box==status2.on_box\

            and status1.hold == status2.hold:

                return True

        else:

            return False


def print_status(status, database):

    if database[status.father].father != -1:

        print_status(database[status.father], database)

        father_status = database[status.father]

        print('if (' + father_status.loc_monkey + ',' + father_status.loc_box
+ ',' + father_status.loc_banana + ',' + str(father_status.on_box) + ','
+ str(father_status.hold) + \

            ') then (' + status.loc_monkey + ',' + status.loc_box + ',' +
status.loc_banana + ',' + str(status.on_box) + ',' + str(status.hold) + ')')

        return


def add(database, new_status):

    flag = True

    for new in new_status:

        for old in database:

            if is_equal(new, old):

                flag = False

                break

    if flag:

        database.append(new)

```

```
return database
```

```
def init_database():
```

```
    database = []
```

```
    init_status = status()
```

```
    database.append(init_status)
```

```
    return database
```

```
def main():
```

```
    final_status = status('c','c','c',0,0)
```

```
    database = init_database()
```

```
    already_checked = 0
```

```
    is_found = False
```

```
    # 如果 database 中还有未 check 的事实，就拿出一个，求它产生的新状态，加入 database 中
```

```
    while already_checked < len(database):
```

```
        now_status = database[already_checked]
```

```
        # 得到 goto 能够产生的状态
```

```
        new_status = now_status.goto(already_checked)
```

```
        # 判断 new_status 中的状态是否已存在，不存在则加入 data_base
```

```
        database = add(database, new_status)
```

```
    for item in database:
```

```
        if is_equal(item, final_status):
```

```
            is_found = True
```

```
            final_status.father = item.father
```

```
            break
```

```
    if is_found:
```

```
        break
```

```

# 得到 push 能够产生的状态

new_status = now_status.push_box(already_checked)

database = add(database, new_status)


for item in database:

    if is_equal(item, final_status):

        is_found = True

        final_status.father = item.father

        break

if is_found:

    break


already_checked += 1


if is_found:

    print_status(final_status,database)

    print("if (c,c,c,0,0)  then (c,c,c,1,0)\nif (c,c,c,1,0)  then
(c,c,c,1,1)")

else:

    print('Not found.')


if __name__ == '__main__':

    main()

```