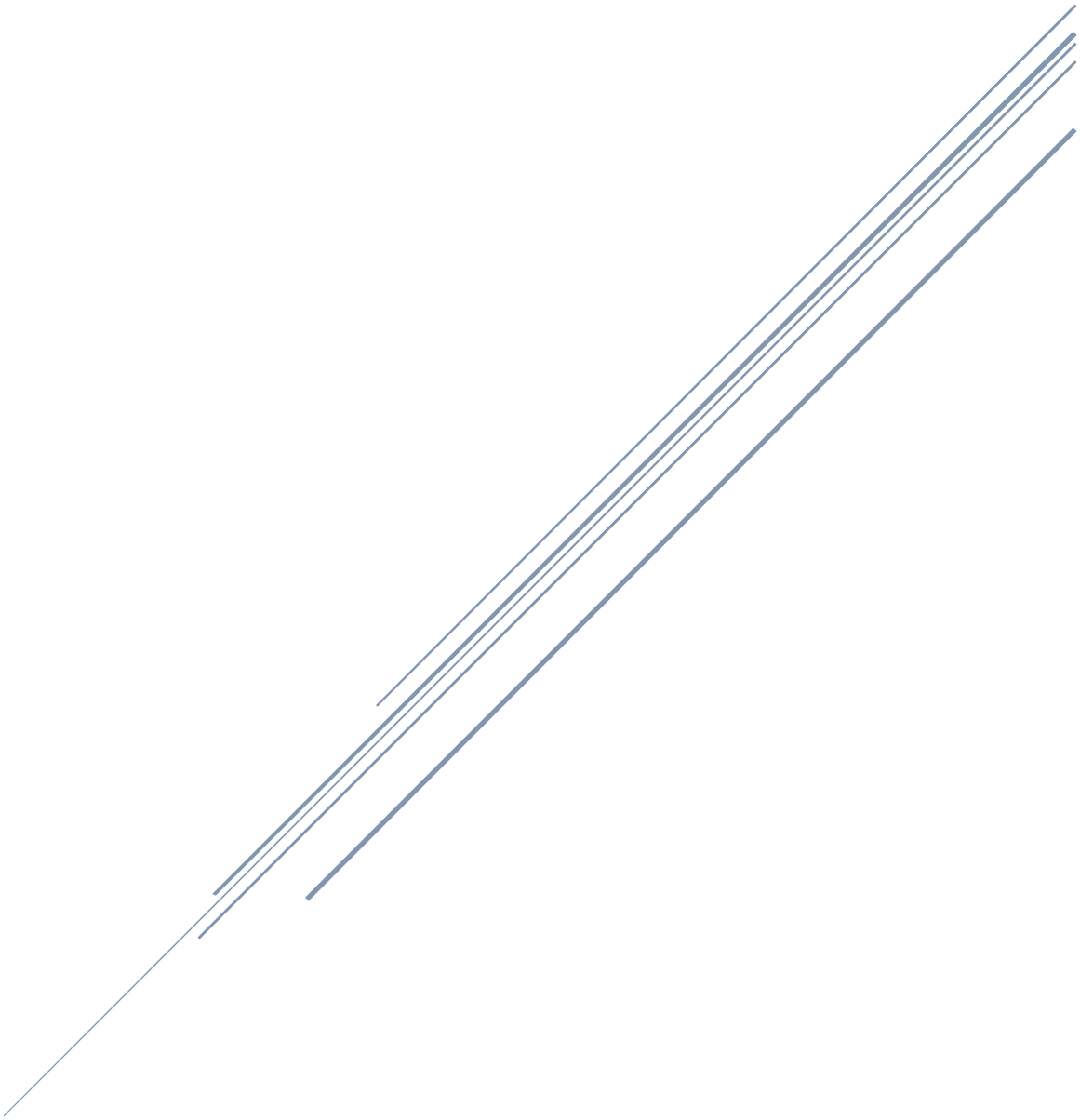
BLOCKS WORLD REPORT

Chappell Jones



Wright State University
CS 4850

ABSTRACT

The purpose of this report is to provide insight for my Blocks World program. It also explains the methodology of stacking the blocks in the sequential manner until it reaches the desired state as shown in Chapter 11. The changes in the experiments will show the states going from one state to another while displaying commands and relations after a block is manipulated. The flow chart will provide visual aide of the operation of the process using a loop for the overall operation, another loop containing the decision phase for picking and place the blocks according to the requirements to reach the desired result of the final state.

Table of Contents

INTRODUCTION

The Blocks World project was done for the sole purpose of emulating the changing of location of each block according to what the content of the final state. Not only the program will show each different state of change, but also report on the different commands manipulating certain blocks and print what location the robot hand is at and the next location the robot hand will go. The number of states will be printed alongside the list of commands for the state and relations for the block. This feature is to keep track of the number of possible steps for the blocks going from initial state to final state.

Goals

The goal of Blocks World is to get a report of how many steps it takes to get from an initial state to final state and to show that even though the program may be inefficient and that it takes a long time, the program still works. Like in Chapter 11 “States” another goal is to get the number of states, all the possible commands for each state at a time and relations of one block as it is moved from one location to another. The results will have two trials of inputs to confirm that the program works. The results will only have the first three stages from the initial state and three states before the final state because of the large length of the process.

Results

Trial 1

Enter values for Location 1:ca

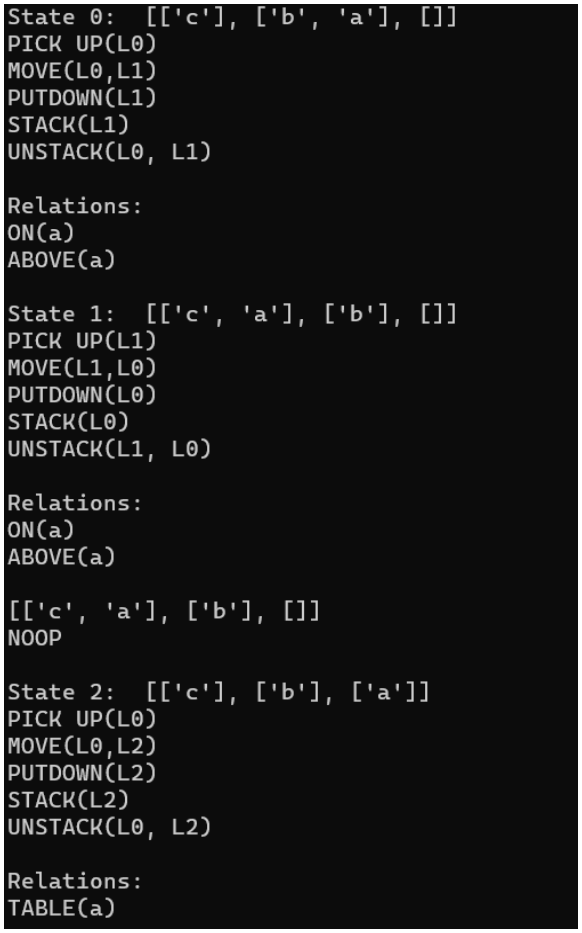
Enter values for Location 2:b

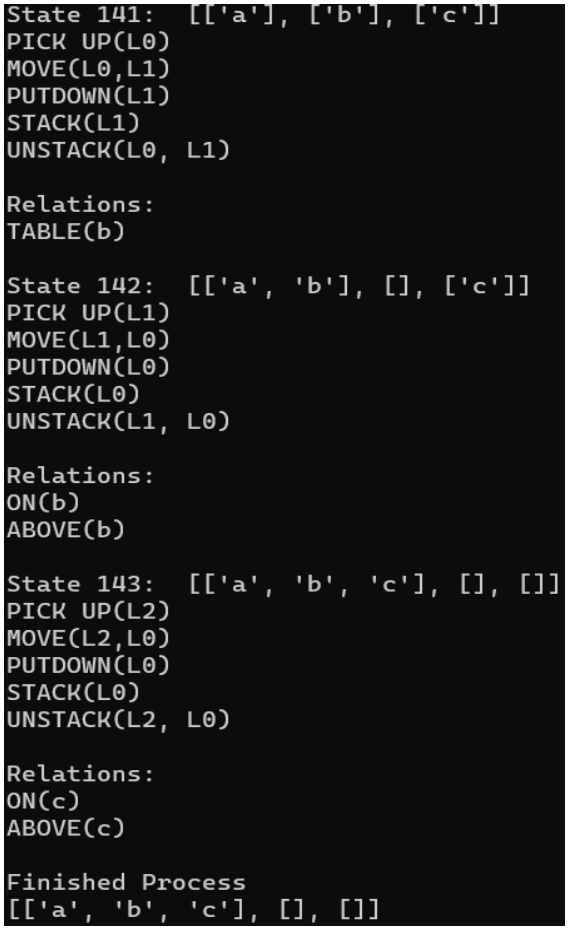
Enter values for Location 3:

Enter values for output location 1:abc

Enter values for output location 2:

Enter values for output location 3:





Trial 2

Enter values for Location 1:ca

Enter values for Location 2:b

Enter values for Location 3:

Enter values for output location 1:abc

Enter values for output location 2:

Enter values for output location 3:

```
State 0:  [['c'], ['b', 'a'], []]
PICK UP(L0)
MOVE(L0,L1)
PUTDOWN(L1)
STACK(L1)
UNSTACK(L0, L1)

Relations:
ON(a)
ABOVE(a)

State 1:  [['c', 'a'], ['b'], []]
PICK UP(L1)
MOVE(L1,L0)
PUTDOWN(L0)
STACK(L0)
UNSTACK(L1, L0)

Relations:
ON(a)
ABOVE(a)

[['c', 'a'], ['b'], []]
NOOP

State 2:  [['c'], ['b'], ['a']]
PICK UP(L0)
MOVE(L0,L2)
PUTDOWN(L2)
STACK(L2)
UNSTACK(L0, L2)

Relations:
TABLE(a)
```

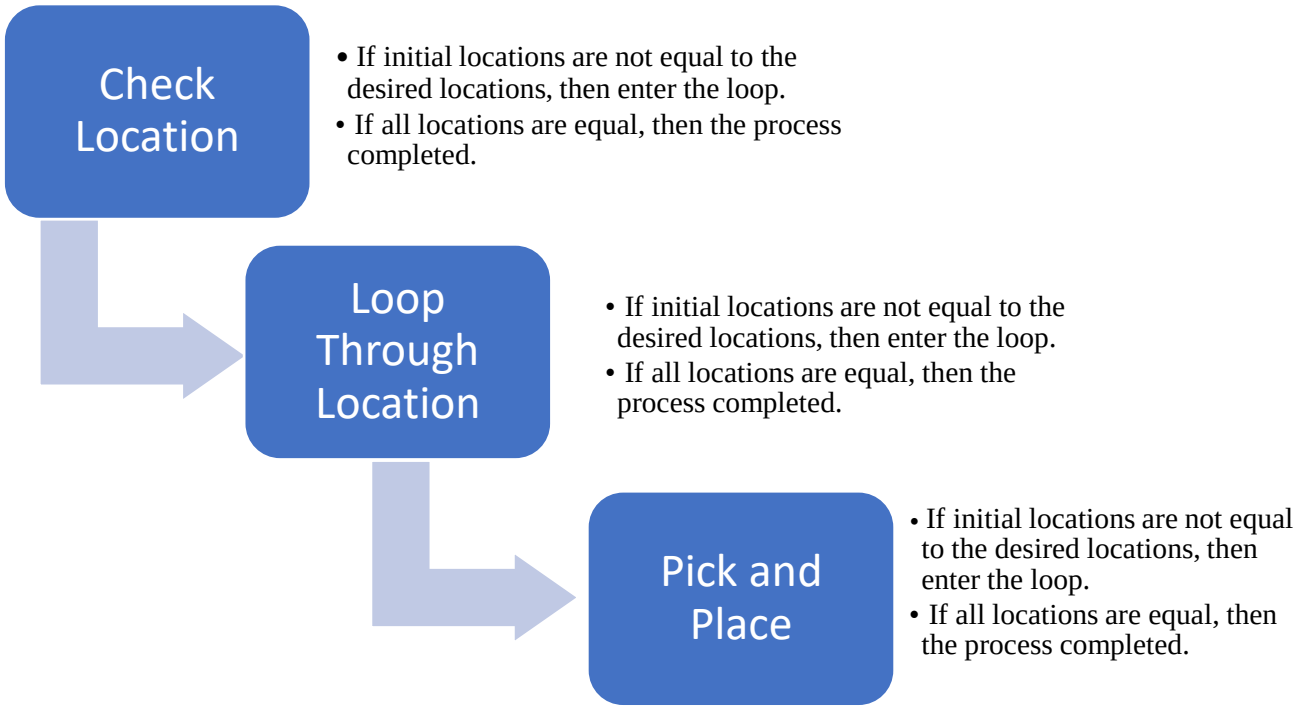
```
State 71:  [['b', 'a'], [], ['c']]
PICK UP(L1)
MOVE(L1,L0)
PUTDOWN(L0)
STACK(L0)
UNSTACK(L1, L0)

Relations:
ON(a)
ABOVE(a)

State 72:  [['b', 'a', 'c'], [], []]
PICK UP(L2)
MOVE(L2,L0)
PUTDOWN(L0)
STACK(L0)
UNSTACK(L2, L0)

Relations:
ON(c)
ABOVE(c)

State 73:  [['b', 'a'], ['c'], []]
PICK UP(L0)
MOVE(L0,L1)
PUTDOWN(L1)
STACK(L1)
UNSTACK(L0, L1)

Relations:
TABLE(c)

Finished Process
[['b', 'a'], ['c'], []]
```

Methodology

Process

First the user will input blocks for the initial state in all three locations. Next, the user will be prompted to enter the blocks for the final state. After the inputs and outputs are entered, the concatenation of the strings entered will take place and fill the letters in the chosen locations. The three locations of the states are represented in stack. The leftmost character represents the bottom of the stack, and the rightmost character represents the top of the stack. If all locations are not equal to the final state, then enter the while loop to proceed with the operation within the while loop. If all locations are equal, then the process is completed. If a stack is empty, then the robot hand will do nothing. When a non-empty location is detected, a decision of where to put a block will be made. The blocks will be moved and placed based on the decided location. Once the placing of the block is finished, the status will print, and the process will start again until the final stage is reached.

Flow Chart



Conclusions

The overall project did not go as expected when I entered a, b, c, d, e, f, g, h, i, j, m, n inputs and outputs. When I ran those inputs and outputs the program kept running for a long time, exceeding 10 minutes. This is mainly because of the part where the decision of which location to choose is based on choosing a location at random other than the current location. This approach was inefficient but, the program does what I need it to do when I provide smaller amounts of blocks to move. What would have made my program more efficient is to have it go through the final state locations one block at a time, unstack until the top of the desired block is clear and put that block in its proper location, then start the process all over again until initial state locations and final state locations are equal.

Resources

Chapter 11 State Changes PowerPoint slides

Chapter 11b A.I. Project Example