World of Blocks Solver
Foundations of Artificial Intelligence

Ethan Heinlein
11/27/2023

Tech Choices

- Python 3.12
  - Helpful features
    - list[-1] to get item at end of list
    - No worries about re-casting when assigning
    - No worries about garbage collection
- Tkinter GUI Library
- Multiprocessing Library
- Copy Library to quickly create deep copies of nodes/actions
- Standard libraries like time and traceback

Definitions

- State: A data structure containing information about:
  - Current block states
  - Current table state
  - Claw position/block held
- Stack: The list of blocks in a particular table spot
- Queue: Standard queue data structure
- Tree: Dictionary containing every node in a **layer**

- Layer: List of all states on same depth level

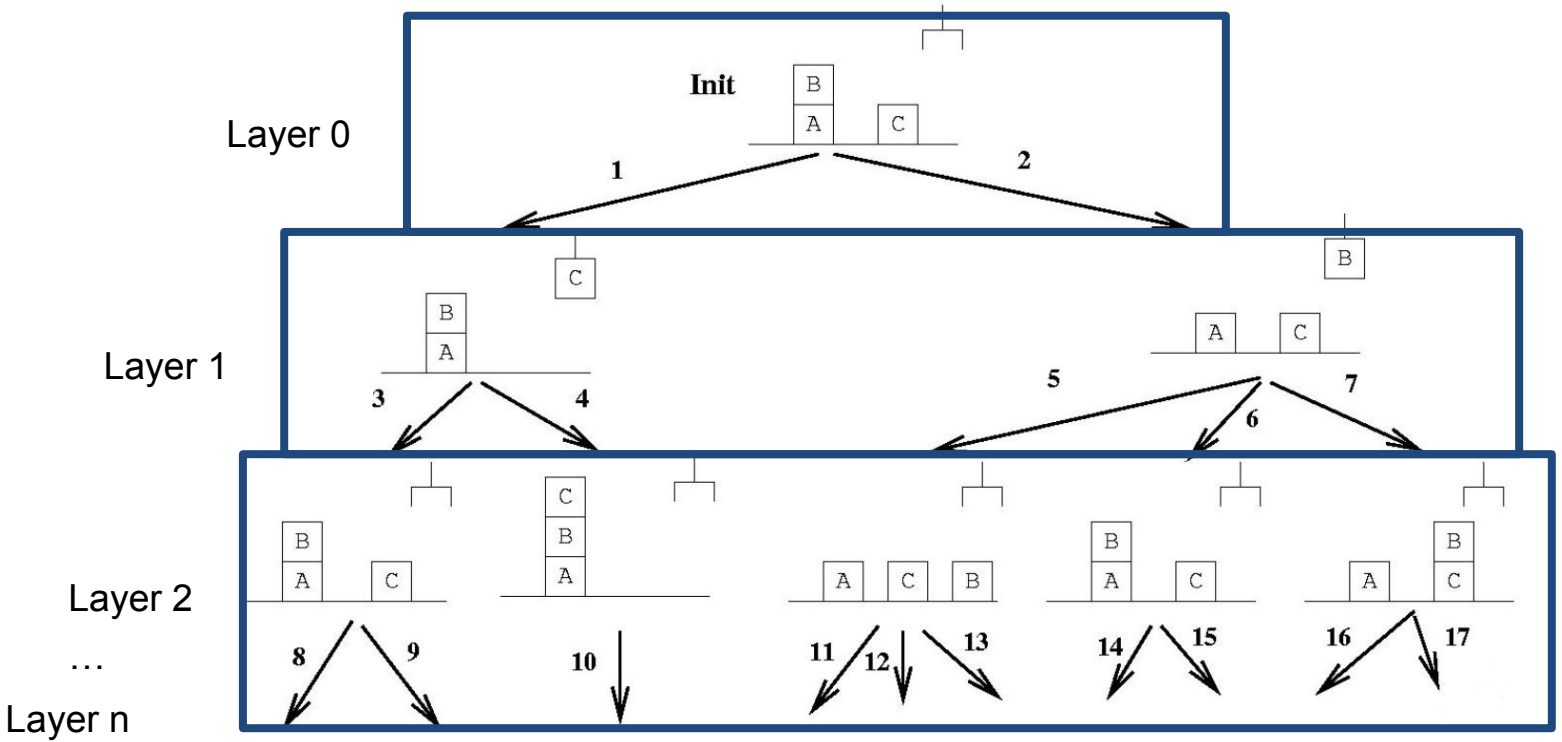

Figure 1: From https://www.cs.bham.ac.uk/~mmk/Teaching/AI/l8.html

Block Relation Definitions

- CLEAR(x) - Nothing is above x
- ABOVE(x, y) - x is above y
- ON(x, y) - x is **directly** on top of y (x.above[-1])
- TABLE(x) - x is **directly** on the table

Action Definitions and Preconditions

- PICK-UP(Li) - A block is picked up from table spot Li
  - Arm is empty
  - stack[Li][-1] is on the TABLE
  - stack[Li][-1] is CLEAR
  - Claw is at location Li
- PUT-DOWN(Li) - Held block placed at table spot Li
  - Arm is holding something
  - Stack at Li is empty (no blocks)
  - Claw is at location Li

- STACK(Li) - Held block placed at stop of stack Li
  - Arm is holding something
  - stack[Li][-1] is CLEAR
  - stack[Li][-1] is **not** on the TABLE
  - stack at Li is **not** empty
- UNSTACK(Li) - Block picked up from stack at spot Li
  - Arm is empty
  - stack[Li][-1] is CLEAR
  - stack[Li][-1] is **not** on the TABLE
  - stack at Li is **not** empty
- MOVE(Li, Lk) - Claw moved from spot Li to Lk
  - Claw isn't at Lk already (Li != Lk)
  - Lk is a legal spot (1, 2, 3)

Methodology

Breadth-First FIFO

Green's Method

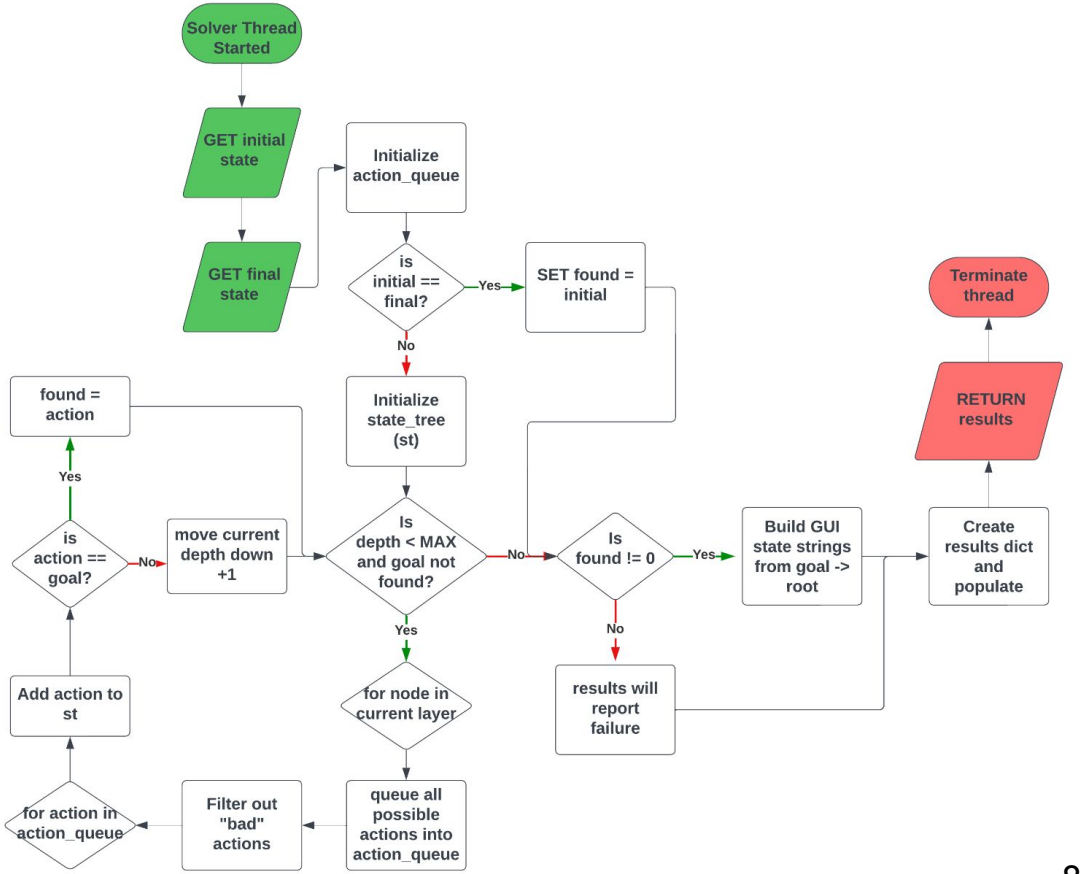

Figure 2: From Ethan Heinlein

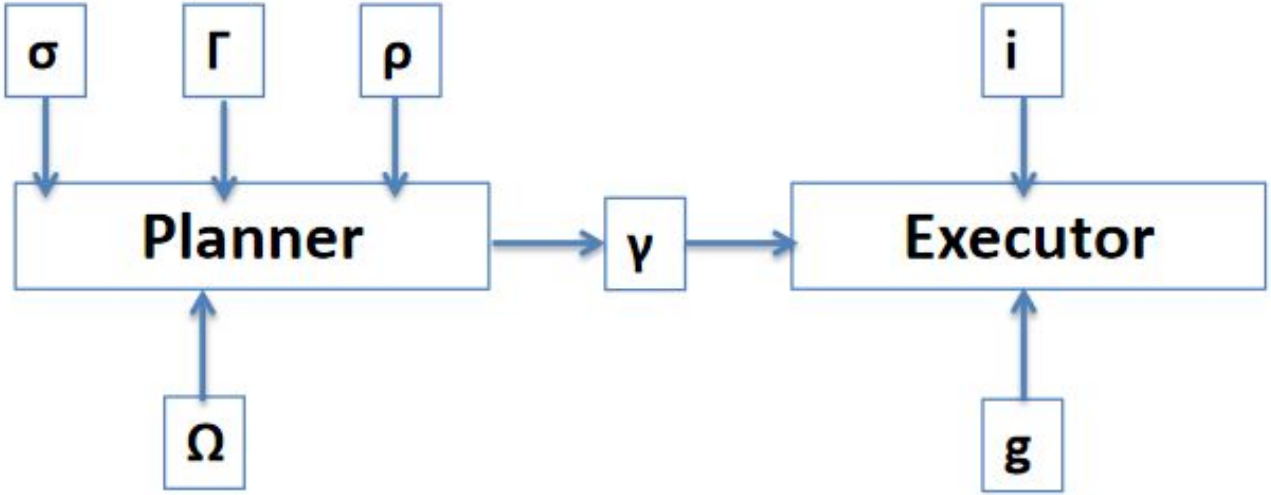

Figure 3: From *Logical Foundations of Artificial Intelligence* Genesereth and Nilsson

Actions loaded into database (**Ω**). Designator (**⌈**) created by dequeuing from database. Passed to Executor (**γ**) and compared against goal state (**g**).

How do we know if two states are equal?

```
class Table_State:
    def __eq__(self, other):
        res2 = self.__compare_stack(self.L2, other.L2)
        res3 = self.__compare_stack(self.L3, other.L3)
        if res1 and res2 and res3:
            return True
        return False

    def __compare_stack(self, stack, other):
        if len(stack) != len(other):
            return False
        # Check if both are empty
        if len(stack) == 0 and len(other) == 0:
            return True
        for i in range(0, len(stack)):
            # Compare the blocks above this block
            above_me = stack[i:len(stack)]
            above_other = other[i:len(stack)]
            if above_me != above_other:
                return False
            if stack[i] != other[i]:
                return False
        return True
```

```
class Block_State:
    def __eq__(self, other): # Overwrite to compare two blocks to each other
        # Account for other being null
        if(not other):
            return False
        # Check single properties
        if self.label != other.label or self.table != other.table \
                or self.clear != other.clear:
            return False
        # All tests passed
        return True
```

Optimizations Made

- “Inverse” actions of the parent are filtered out
  - Ex: Parent: M(1, 3) Child: M(3, 1)
- The GUI and Solver are fairly detached, communicating only with formatted strings of states
- Parallelism implemented to improve overall program performance

Room for Improvement

- Further filtering to reduce redundant states
  - Prevent double moving?
- Algorithm is **not** fully optimized
  - Lots of redundant states taking up computation time
- Green's Method not totally optimal

References

- [1] M. R. Genesereth and N. J. Nilsson, Logical Foundations of Artificial Intelligence. Morgan Kaufmann, 1987.
- [2] A. Dixit, “Goal stack planning for blocks world problem,” Medium, https://apoorvdixit619.medium.com/goal-stack-planning-for-blocks-world-problem-41779d090f29 (accessed Nov. 27, 2023).
- [3] “The blocks world,” Introduction to AI - Week 8, https://www.cs.bham.ac.uk/~mmk/Teaching/AI/l8.html (accessed Nov. 27, 2023).
- [4] “Search algorithms in ai,” GeeksforGeeks, https://www.geeksforgeeks.org/search-algorithms-in-ai/ (accessed Nov. 27, 2023).

Thank you

See project README file for more information