

Natural Language Interface w/ Cozmo: GPT3-Powered Planning

Rohan Patel

Programming w/ Finite State Machines



- Using cozmo-tools, we can program Cozmo to perform actions using a Finite State Machine
- simpler/more abstract than Python code, yet still require syntax knowledge

Greet.fsm

```
"""
    The Greet demo illustrates the use of CompletionTrans and TimerTrans
    transitions.

    Behavior: Cozmo starts out by saying 'Greetings, human!'. After his
    speech has completed, he waits 5 seconds, then says 'Bye-bye now'.
    """

from cozmo_fsm import *

class Greet(StateMachineProgram):
    $setup {
        say: Say('Greetings, human!') =C=>
        wait: StateNode() =T(5)=>
        say2: Say('Bye-bye now.')
    }
```



Using GPT-3.5 to Generate FSMs

- Want to create a simple command-line interface for communicating with Cozmo, powered by GPT-3.5
 - Using model gpt3.5-turbo like in class, but show it plenty of examples
 - Ideally can parse complex logic from the user's specifications

- Experimented with both rules-based and example-based prompting (i.e, tell it how to make FSM vs. just show it FSMs)
 - Found that the latter approach works better from a syntax perspective

Stage I: Code Generation

Preamble: “You generate FSM files, according to the user’s specifications.”

Premise: I fed the model all the examples in `cozmo_fsm`

- Contains many examples w/ annotations of function in comments

Query: “Move forward by 100 centimeters.”

```
response = openai.ChatCompletion.create(  
    model="gpt-3.5-turbo",  
    messages = messages  
)
```

```
rohanp@nori:~/15-494/final$ python3 code/generate.py  
What would you like Cozmo to do? Move forward by 100 centimeters  
This is a simple behavior that moves Cozmo forward by 100 cm:  
...  
from cozmo_fsm import *  
  
class MoveForward(StateMachineProgram):  
    $setup{  
        move: Forward(100)  
    }  
...
```

Stage II: Instant Program Execution

```
When you run this program, Cozmo will move forward by 1 meter, or 100 centimeters.  
Writing code to new folder: tmp/generated/2023-04-2813:34:58.760261/ ...  
Wrote generated code to tmp/generated/2023-04-2813:34:58.760261/MoveForward.py.  
running ...
```

- Want to save file, convert to .py with *genfsm*, and execute directly on Cozmo
- Running the last two tasks using **subprocess**
- Experimented with *simple_cli*, but very slow
 - Wrote some helper code to use cozmo directly

```
running_fsm = cozmo_fsm.program.running_fsm = the_class()  
  
cli_globals = globals()  
cli_globals['running_fsm'] = running_fsm  
  
robot.loop.call_soon(running_fsm.start)  
  
while True:  
    time.sleep(30)  
  
return running_fsm  
  
if __name__ == "__main__":  
    global mod_name  
  
    mod_name = sys.argv[1]  
    path = sys.argv[2]  
  
    sys.path.append(path)  
  
    print("module name: " + mod_name)  
    cozmo.run_program(fsm_replacement)
```

More Complex Logic

What would you like Cozmo to do? If you see a cube, turn left by 90 degrees, else turn right by 90 degrees
Here's the code to make Cozmo turn left by 90 degrees if the robot sees any cube, else turn right by 90 degrees:

```
```python
from cozmo_fsm import *

class CubeAware(StateMachineProgram):
 $setup {
 look: AnimationTrigger('anim_speedtap_lookgetin_01')
 =T(1)=> detect: DetectVisibleObject(robot.world.light_cubes.values(),False)
 =T(1)=> {found,not_found}

 found: Turn(-90) =C=> finished: Say('I found a cube')

 not_found: Turn(90) =C=> finished: Say('I did not find a cube')

 {found,not_found} =C=> look
 ... }
```
```

The function starts by launching an animation trigger that makes Cozmo look around for cube-shaped objects. After a one-second of the robot's light cubes are visible to the camera. If a cube is found, Cozmo turns left by 90 degrees and speaks "I found a cube". If no cube is found, Cozmo turns right by 90 degrees and speaks "I did not find a cube". In either case, the program finishes by restarting the "look" state node to resume scanning for cubes.

Unexpected Behavior

I just prompted it to spin in a circle:

```
class SpinInCircle(StateMachineProgram):  
    $setup{  
        spin: Turn(720) =C=> Say("I'm dizzy!")  
    }
```

Future Directions

- Better prompting to improve robustness and quality of responses
 - Larger example set, input sanitation, more clear prompts/instructions, edge cases
- GPT's API is the speed bottleneck - need to improve latency
 - Use a smaller, faster, and/or more specialized model (local?)
- Enable voice control
 - Using OpenAI Whisper, incorporate Automatic Speech Recognition (ASR) to transcribe voice commands in real-time, feed directly into GPT-3.5 and execute FSM on Cozmo
 - Likely to significantly slow down program execution
- GPT-4??

