Animating Parsing :

Finite State Machine, Parsing Tree, Earley's Parse Animation

Jessica Lei, Mingnan Su

Problem And Motivation

Problem:

The problem of using Jupyter notebook is that users have the ability to present algorithms along with the corresponding code segments, but no visualization.

Motivation:

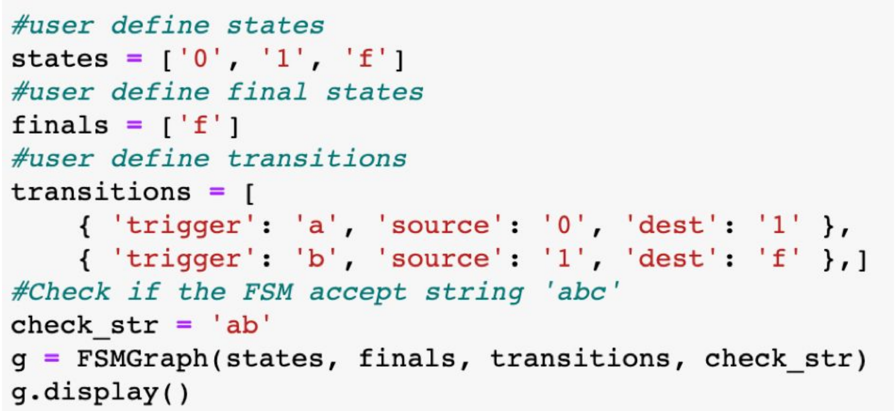
Our project aims to add animated graph visualization to the execution of finite state machines, derivation of sentences and algorithm of Earley's parser animation.

Finite State Machine Animation

Step 1: Let the user input states and transitions

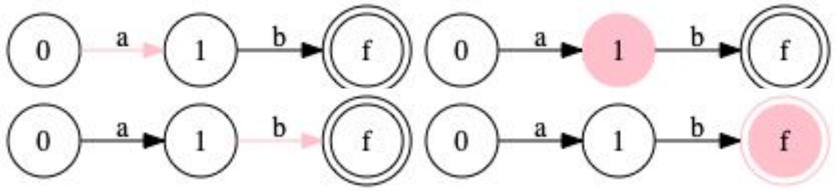
Step 2: Let the user input the string

Step 3: Fsm is going to check if it accepts it



```
#user define states
states = ['0', '1', 'f']
#user define final states
finals = ['f']
#user define transitions
transitions = [
    { 'trigger': 'a', 'source': '0', 'dest': '1' },
    { 'trigger': 'b', 'source': '1', 'dest': 'f' },]
#Check if the FSM accept string 'abc'
check_str = 'ab'
g = FSMGraph(states, finals, transitions, check_str)
g.display()
```

Step 4: Animate the execution process

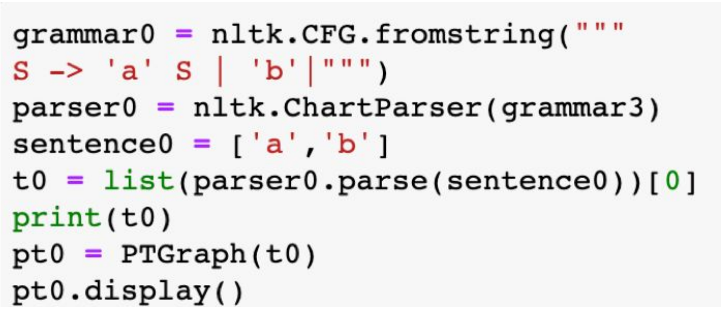


Sentence Derivation Animation

Step 1: Let the user input the grammar

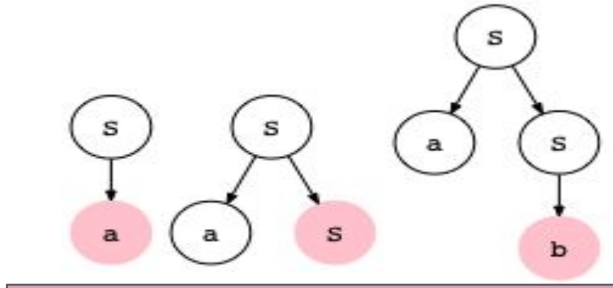
Step 2: Let the user input the sentence

Step 3: Parse the sentence into the grammar



```
grammar0 = nltk.CFG.fromstring("""
S -> 'a' S | 'b'|""")
parser0 = nltk.ChartParser(grammar3)
sentence0 = ['a','b']
t0 = list(parser0.parse(sentence0))[0]
print(t0)
pt0 = PTGraph(t0)
pt0.display()
```

Step 4: Animate the execution process

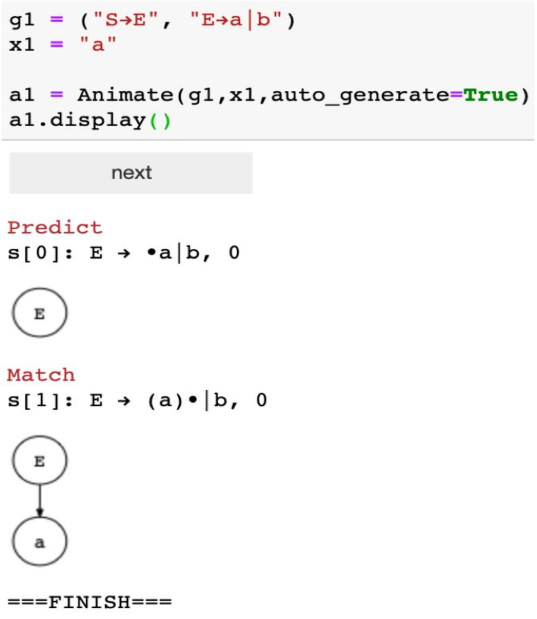


Earley's Parse Animation

Step 1: Let the user input the grammar

Step 2: Let the user input the sentence

Step 3: Parse the sentence into the grammar



```
g1 = ("S→E", "E→a|b")
x1 = "a"

a1 = Animate(g1,x1,auto_generate=True)
a1.display()
```

```
Predict
s[0]: E → •a|b, 0
```

```
Match
s[1]: E → (a)•|b, 0
```

```
===FINISH===
```

Implementation

Test

- Conducted tests inside the Jupyter notebook. Passing in the correct parameters to check the output by observing the animation graph.
- Unit testing and black box testing

Documentation

Documented this project manually inside Documentation.ipynb, where it contains the description of each class and method. Some reasoning behind the modification of earley's parser from lecture notes is documented as well.

Statistics

- Size of Code: 661 Lines
- Size of Test Cases: 200 Lines

Conclusion

- This project is useful to visualize the data flow of a given algorithm or parser. However our visualization techniques depend on the parser of the given algorithm.
- Challenges we met are constructing Finite State Machine transitions, derivation of sentence grammar

References

1.Zuzak, Ivan, and Vedrana Jankovic. "FSM Simulator." FSM Simulator, ivanzuzak.info/noam/webapps/fsm_simulator/.

2.Hasebe, Yoichiro. "RSyntaxTree." Yohasebe.com, yohasebe.com/rsyntaxtree/.