Python DPDA Executor

Lyell Read, CS321H / F2020

High Level Idea

- Represent the components of a DPDA using simple data structures that are already present in Python
- Execute the acceptance or denial of a string in a provided DPDA
- Create DPDA Complement
- Execute the acceptance or denial of a string in the complement of the provided DPDA
- Neat debug printout for user understanding.

Data Structures?

- D**P**DA Requires a **Pushdown Stack.**
  - Use a list in python:
    - Pop: list.pop()
    - Push: list.append()
  - Intricacy: Must push things in proper order, i.e. ‘aZ’ must be pushed as ‘Z’, ‘a’.
- DPDA Requires a set of transitions
  - Nested lists
    - [[q0, x, y, q1, z], [...], ...]
- How to represent the whole DPDA for input?
  - JSON
- How to represent the whole DPDA internally?
  - Dictionary, easily parsed by json.loads()

JSON DPDA Representation

How the program works

1. Parse the JSON, sanity check the start and end states.
2. Loop:
    a. Filter the list of all transitions by which one(s) we can take given
        i. Current state
        ii. Current character
        iii. Stack top character
    b. Take the move, provided there is only one transition
        i. If there are no possible transitions, we are at a trap / ‘end’ state
        ii. If there are multiple transitions, this signals nondeterminism.
3. Repeat for language / DPDA Complement

DPDA Complement?

To get the complement of a DPDA, make all non-final states final states, and make all final states non-final states.

For a DPDA represented as a dictionary, this is a matter of simple set manipulation.

```
accepting_states = dpda_dict["final_states"]
nonaccepting_states = (
    set(dpda_dict["states"])
    - set(dpda_dict["final_states"])
    - {dpda_dict["start_state"]}
)

complement_dict = dpda_dict
complement_dict["final_states"] = list(nonaccepting_states)

return complement_dict
```

Demonstration