Roshan Shetty 13

Compiler Mini Project

Aim: Design a predictive parser for a given language

Code:

```
class PredictiveParser:
	def __init__(self):
		# self.non_terminals = list(input("Enter the list of non-terminals >"))
		# self.terminals = list(input("Enter the list of terminals >"))
		# print("Use `@` for denoting upsilon.")

		# rule_count = int(input("Enter the number of rules you want to add > "))
		# self.production_rules = list()
		# for i in range(rule_count):
		#	self.production_rules.append(input(f"Enter rule {i + 1} > ").replace(" ", ""))

		# self.first = self.follow = dict()
		# for non_terminal in self.non_terminals:
		#	self.first[non_terminal] = list(input(f"Enter first({non_terminal}) > "))

		# for non_terminal in self.non_terminals:
		#	self.follow[non_terminal] = list(input(f"Enter follow({non_terminal}) > "))

		self.non_terminals = list("EGTUF")
		self.terminals = list("+*()a")
		self.production_rules = ["E->TG", "G->+TG", "G->@", "T->FU", "U->*FU", "U->@", "F->(E)", "F->a"]
		self.first = {"E":["(", "a"], "G":["+", "@"], "T":["(", "a"], "U":["*", "@"], "F":["(", "a"]}
		self.follow = {"E":[")", "$"], "G":[")", "$"], "T":[")", "$", "+"], "U":[")", "$", "+"], "F":[")", "$", "+", "*"]}


	def generate_parsing_table(self) -> dict[str, list[str]]:
		parsing_table = dict()
		for non_terminal in self.non_terminals:
			parsing_table[non_terminal] = [None for i in range(len(self.terminals) + 1)]
		for production_rule in self.production_rules:
			non_terminal_at_left, remainder = production_rule.split("->") if "->" in production_rule else production_rule.split("-")
			if not (remainder[0].isupper() or remainder[0] == "@"):
				parsing_table[non_terminal_at_left][self.terminals.index(remainder[0])] = production_rule
			else:
				update_locations = self.first[non_terminal_at_left]
				if "@" in update_locations:
					update_locations.remove("@")
					update_locations += self.follow[non_terminal_at_left]
```

```
                                for update_location in update_locations:
                                        try:
                                                position = self.terminals.index(update_location)
                                        except ValueError:
                                                position = len(self.terminals)
                                        parsing_table[non_terminal_at_left][position] =
production_rule

                return parsing_table

        def print_parsing_table(self, parsing_table : dict[str, list[str]]):
                print("Non Terminal", end = "\t")
                for terminal in self.terminals:
                        print(terminal, end = "\t")
                print("$", end = "\n")

                for entry in parsing_table:
                        print(entry, end = "\t")
                        for cell in parsing_table[entry]:
                                print(cell, end = "\t")
                        print(end = "\n")

if __name__ == '__main__':
        predictive_parser = PredictiveParser()
        parsing_table = predictive_parser.generate_parsing_table()
        predictive_parser.print_parsing_table(parsing_table)
```

Output:

```
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
Type "copyright", "credits" or "license" for more information.

IPython 7.20.0 -- An enhanced Interactive Python.

In [1]: runfile('/home/[redacted]/Desktop/predictiveparser.py', wdir='/home/[redacted]Desktop')
Non Terminal    +   *   (   )   a   $
E   None    None    E->TG   None    E->TG   None
G   G->@    None    None    G->@    None    G->@
T   None    None    T->FU   None    T->FU   None
U   U->@    U->@    None    U->@    None    U->@
F   None    None    F->(E)  None    F->a    None

In [2]:
```