Année Universitaire 2023/2024

Rapport de Projet :

Sujet :

Génération de l'implantation procédurale directe d'un automate à états finis

- Réalisé par :

  Oumaima El Hadraoui

  Fatimazahrae Imloul

- Encadré par :

  Mr. KABBAJ ADIL

Introduction :

Afin d'assimiler le cours de techniques de compilation, les projets en générale sont des outils pour mettre en pratique la connaissance théorique prise dans la classe, d'où le but de notre sujet ; la Génération de l'implantation procédurale directe d'un automate à états finis.

Analyse de sujet :

Le but de notre sujet est de réalisé un programme sous python qui reçoit en entrée un automate à état fini en tant que graphe et produit en sortie sa représentation procédurale directe.

Entrée :

- **Automate à état finis :**

  Automate A = (T, E, D, ei, Ef)

  T = Ensemble des unités terminales

  E = Ensemble des états possibles d'Automate A

  D = Ensemble des règles de transition

  ei = Etat initial

  Ef = Ensemble des états finaux

Sortie :

- **<u>Implantation procédurale directe d'un automate à état finis</u>** :

```
void Etat1()
{
   char c;
   c= getchar();
   swicth(c) {
      case 'a' : Etat2();
                 break;
      case 'b' : Etat5();
                 break;
      case 'c' : Etat57();
                 break;
      ...
      default : ERREUR();
   }
 }

 void Etat36524()
 ...
```

Etape 01 : Création de la classe Automate et initialisation de ses attributs

- Ensemble des Etats d'Automate (Liste)
- Ensemble des transitions d'Automate (Dictionnaire)
- Etat initial
- Liste des états finaux
- Les unités terminales (Chaine de caractère)

```
1
2 class Automate:
3     def __init__(self, alphabet):
4         self.states = []
5         self.transitions = {}
6         self.initial = None
7         self.finals = []
8         self.alphabet = set(alphabet) #alphabet est l'ensemble des unites terminals
9
```

Etape 02 : L'ajout d'un état

```
11     def add_state(self, state, final=False):
12         if state in self.states:
13             print("error : state " + state + " already exists.")
14             return
15         self.transitions[state] = []
16         self.states.append(state)
17         if final:
18             self.finals.append(state)
```

Etape 03 : Ajouter Transition

-La 1ière fonction renvoie vrai si le symbole fait partie de l'alphabet, faux sinon.

- la 2ième Trouve la destination d'une transition depuis un état source donné, en utilisant un symbole spécifié.

```
20    def valid_symbol(self, symbol):
21        return symbol in self.alphabet
22
23    def destination(self, src_state, symbol):
24        if src_state not in self.states:
25            print("error : the state '" + src_state + "' is not an existing state.")
26            return
27        for (s, destination) in self.transitions[src_state]:
28            if s == symbol:
29                return destination
30        return None
```

- Déclencher erreur d'impression si l'automate possède déjà une transition pour l'état source et le symbole spécifiés.

```
32    def add_transition(self, src_state, symbol, destination):
33        if not self.valid_symbol(symbol):
34            print("error : the symbol '" + symbol + "' is not part of the alphabet.")
35            return
36        if src_state not in self.states:
37            print("error : the state '" + src_state + "' is not an existing state.")
38            return
39        if destination not in self.states:
40            print("error : the state '" + destination + "' is not an existing state.")
41            return
42        if self.destination(src_state, symbol) is not None:
43            print("error : the transition (" + src_state + ", " + symbol + " ) already exists.")
44            return
45        self.transitions[src_state].append((symbol, destination))
```

Etape 04 : Générer la représentation procédurale

```
47    def generate_procedural_representation(self):
48        for state in self.states:
49            print(f"void Etat_{state}()\n"
50                  "{\n"
51                  "    char c;\n"
52                  "    switch(c)\n"
53                  "    {\n")
54            for (sym, dest) in self.transitions[state]:
55                print(f"        case '{sym}': Etat_{dest}();\n"
56                      "                      break;\n")
57            print("        default: ERREUR();\n"
58                  "    }\n"
59                  "}\n")
60
```

Etape 05 : Teste

```python
def main():
    alphabet_input = input("Enter the alphabet: ")
    a = Automate(alphabet_input)

    state_input = input("Enter a state: ")
    a.add_state(state_input)

    final_state_input = input("Is it a final state? (y/n): ").lower()
    if final_state_input == 'y':
        a.add_state(state_input, True)

    initial_state_input = input("Enter the initial state: ")
    a.init = initial_state_input

    while True:
        src_state = input("Enter the source state for a transition (or enter 'done' to finish): ")
        if src_state.lower() == 'done':
            break

        symbol = input("Enter the symbol for the transition: ")
        dst_state = input("Enter the destination state for the transition: ")

        a.add_transition(src_state, symbol, dst_state)

    print("\nAutomaton details:")
    print(a)
    print("\nProcedural Representation:")
    a.generate_procedural_representation()


if __name__ == "__main__":
    main()
```

Conclusion :

C'était un défait pour nous de mettre en pratique le cours ; car on est rencontrer par pas mal de problèmes en tête d'ils la POO en python ainsi que la compréhension parfaite de sujet ,mais il mérite le coup d'essaye et il est un apport important pour un informaticien.