Création de la méthode afficherAutomate avec PGF/Tikz :

PGF/TikZ

C'est une combinaison de deux langages informatiques pour la création de graphiques vectoriels. PGF est un langage de bas niveau, tandis que TikZ est un ensemble de macros qui fournit une syntaxe plus simple comparée à celle de PGF. Il a été créé en 2005 par Till Tantau qui est aussi le développeur principal de l'interpréteur de PGF et TikZ qui est écrit en TeX. PGF est le sigle de Portable Graphics Format, tandis que TikZ est un acronyme récursif de TikZ ist kein Zeichenprogramm. L'interpréteur de PGF/TikZ peut être utilisé depuis les paquets LaTeX.

Le corps des méthodes afficherAutomate() :

Apres avoir fait plusieurs recherches pour trouver une manière d'afficher les deux automates on a trouvé que la méthode la plus simple c'est d'utiliser les flux d'entrées et de sorties dans un fichier, le package Tikz de Latex et les deux bibliothèques de Tikz (Arrows et automata) afin de les afficher. En suivant les étapes suivantes :

Préambule du fichier Latex :

1- On ouvre le fichier en écriture en utilisant le flux ofstream et ça génère un fichier Latex si ce dernier n'existait pas déjà. Et Une fois le fichier ouvert, en utilisant « << » on commence à écrire dans le fichier.
2- D'abord on commence par le préambule du fichier Latex et TikZ étant un package pour LATEX, il s'utilise comme tout autre package, en déclarant \usepackage{tikz} dans le préambule, les deux bibliothèques (Arrows et automata) et on définit le l'épaisseur des flèches et la distance entre les états.
3- La difficulté qu'on a rencontré dans cette partie c'était le positionnement des états d'une façon que l'automate soit le plus clair et lisible possible. Et après plusieurs tests on a trouvé que le meilleur moyen pour afficher l'automate d'une telle façon est de positionner les états afin d'obtenir un Hexagone et que le nombre d'états est au maximum 10.

Entrée en matière du fichier Latex :

L'élément de base que permet de créer TikZ dans un document LATEX est une figure (picture). Elle se matérialise dans le document LATEX par un environnement tikzpicture.

À l'intérieur de cet environnement se trouve une zone de texte dans laquelle on écrit suivant la syntaxe spéciale de TikZ.

Commençons par dessiner les nœuds. Ses derniers peuvent être positionnés manuellement ou par rapport à d'autres nœuds (positionnement relatif). Et c'est ce qu'on va utiliser dans notre cas.

```
\node[<options>] (name) {text label};
```

1. Les options (dans le contexte des FSM) sont:
- state: **Doit** être spécifié pour créer un état.
- initial: Spécifie un état de départ.
- accepting: Indique l'état final de la machine. (Dessine un double cercle autour de l'état)

2. Le nom (name) est le nom que l'on choisit pour l'état, et pour le remplir on parcourt un ensemble d'état selon le type d'état en utilisant l'accesseur de la classe Etat. Par exemple pour état de départ on va dans l'ensemble etatInitial.
3. Le Text label c'est l'étiquète, et pour le remplir on utilise un entier qu'on incrémente à chaque fois.

Positionnement pour les deux automates (déterministe et non déterministe) :

La position des états peut facilement être spécifiée en rajoutant entre le nom et options :

```
[nom du premier etat position of nom du deuxième etat]
```

Pour positionner les mots-clés sont : right, left, above, below…etc. Ces options peuvent également être combinées.

Afin d'avoir la forme qu'on veut (Hexagone) on a déclaré un tableau de positionnement de 3 case contenant les mots-clés de positionnement suivants : below right, below left et below.

- **Etat initial :**

  Vu que c'est le premier état on n'a pas besoin de spécifier ça position.

- **Les états sauf état final et état initial :**

  On utilise le positionnement relatif et le tableau de positionnement pour les positionner. Pour les deux états qui viennent après l'état initial ils sont toujours positionner en dessous à droite et en dessous à gauche de l'état initial, après vu qu'on utilise des entiers pour les noms des états on a remarqué que dans l'hexagone qu'on cherche à avoir les noms des états sur le côté à droite sont toujours pairs et sur l'autre impairs, et en utilisant cette information on a réussi à avoir l'hexagone qu'on veut.

- **Etat final :**

Pour l'état final on a choisi de le positionner toujours en dessous à droite de l'avant dernier état.

Dessiner les transitions:

Une fois que les états sont tous en place, on commence à ajouter les transitions, c'est-à-dire les bords entre les états. La commande **draw** peut être utilisée pour tracer transitions entre les états déjà créés. En suivant la syntaxe suivante :

```
\draw (<source node>) edge[<edge options>] node{<edge label>} (<dest node>);
```

Remarque:

- <source node> et <dest node> sont les noms des états et **PAS les étiquettes**.
- Par défaut, les bords sont droits, afin d'éviter les chevauchements, ils peuvent être pliés à gauche ou à droite, avec les mots-clés **bend left** et **bend right**.
- Les étiquettes peuvent être positionnées au-dessus ou en dessous du bord, avec les mots-clés **above** et **below** dans les options de bord.

- **Automate détérministe** :

On parcourt l'ensemble global des états et l'alphabet et on test si il y a une transition entre un état et un autre en utilisant la fonction de transition, et on utilise l'accesseur de la classe Etat afin d'y mettre le source node (état source) source et le dest node(état destination) et établir la transition.

- **Automate indéterministe** :

On fait quasiment la même chose que l'automate déterministe sauf que la fonction de transition renvoi un vecteur d'états qui faut aussi parcourir afin d'y mettre le source node (état source) source et le dest node(état destination) et établir les transitions.

Génération du fichier PDF à partir du fichier Latex :

Et finalement on utilise les deux commandes :

- monFlux.close();
- system("pdflatex AfficherAutomate.tex");

La première pour fermer le fichier une fois l'écriture était finie et la deuxième c'est pour générer le fichier PDF.