Déterminisation automatisé d'un AFN

Déroulement

Initialisation

Depuis le module automate `init_aut`
Plusieurs choix : 1. **Par défaut** 2. **Depuis un fichier**

Affichage de l'afn

Présente l'automate devant être déterminiser avec `aff_aut` dans le module afficheur qui utilise `aff_ens` afin d'afficher les éléments depuis une structure **ensemble** défini dans le module ensemble

Determinisation

Dans le module automate `det_aut` copie l'automate en effectuant les changements afin d'obtenir un automate deterministe, pour cela on utilise également `res_trans_d` qui résout le delta des transitions de l'automate déterministe et `tr_finaux` qui inscrit en tant qu'ensemble final ceux contenant un état final de l'afn.

Référencement des ensembles

Dans l'automate déterministe les transitions peuvent se diriger plusieurs fois vers un même ensemble d'états, c'est pourquoi ne seront noté que les indices pointant vers l'ensemble voulu à la place de ceux-ci.

Affichage de l'afd

Structures

```
typedef struct {
  int ens[50];
} ensemble;
```

```
typedef struct {
  ensemble q;
  char a[50];
  int t[50];
  ensemble i;
  ensemble f;
} Automate;

typedef struct {
  ensemble qd[50];
  char a[50];
  ensemble td[50];
  ensemble id;
  ensemble fd[50];
} Automate_d;
```

Référencement d'ensemble Un ensemble est transformé en entier
Tableau de transitions

alphabet	na	a	b
	1	1,2	1
dep	2		3
	3	3	3
		arr	

Éléments du programme

Module ensemble

```
int eg_ens(ensemble ens1,ensemble ens2)
// 1 si ens1 = ens2

int est_ens(ensemble q[],ensemble e)
// 1 si e appartient à la liste d'ensemble q

int est_etat(ensemble e,int n)
// 1 si l'état n fait partie de l'ensemble e

int aj_ens(ensemble q[],ensemble e)
// 1 si l'ensemble est ajouté la liste
```

```
int aj_etat(ensemble *e,int n)
// 1 si l'état est ajouté à l'ensemble

void vider_ens(ensemble *e)
// Vide l'ensemble des données résiduelles précédentes

int etoi(ensemble e,int r[])
// Transforme un ensemble en une liste d'entiers


int trans(int t[],
          int ens_dep,
          char c,
          int ens_arr[])

// * 1 si un état est renvoyé __(ens_arr)__
// * Modifie tableau ens_arr et insère etat d'arrivée des transitions
// depuis etat_dep avec étiquette c
// * Insère seulement à la suite de la table afin d'être conforme
// aux chemins d'un AFN

int est_trans(int t[],
              int etat_dep,
              char etiq,
              int etat_arr)

//* 1 si la transition existe

int aj_trans(int *t,
             int etat_dep,
             char etiq,
             int etat_arr)
```

Module automate

```
void A_defaut(Automate *A)
int A_fichier(Automate *A)
// Insère l'automate configuré en dur dans le code
// ou dans le fichier loader dans A

// pour simplifier l'utilisation on charge directement le fichier
int init_aut(Automate *A)
// Propose à l'utilisateur le choix de la configuration de l'automate
// 1. A_defaut __automate par défaut__
// 2. A_fichier __automate lu dans le fichier *loader*__

int tr_finaux(ensemble f,ensemble q_d[],ensemble *f_d)
// Trouve parmi les ensembles de l'automate déterministe ceux qui sont finaux

int res_trans_d(Automate A,int t_d[][MAX], ensemble q_d[])
// résout le tableau delta des transitions de l'automate
```

```
int det_aut(Automate A,Automate *B)
// Modifie l'automate B pour qu'il soit l'automate A determinisé

int rec_mot(Automate A,char mot[])
// Ecrit reconnu si <mot> est reconnu par l'automate A
```

Prog afficheur

```
void aff_ens(ensemble e)
// Affiche un ensemble d'états e

int aff_trans(int t[])
// Affiche une liste de transitions d'un AFN

int aff_trans_d(ensemble td[])
// Affiche un tableau de transition entre ensembles d'un AFD

int aff_aut(Automate A)
// Affiche un AFN

int aff_aut_d(Automate_d a)
// Affiche un AFD

int main()
// Déclare les automates les initialise, les déterminise et les affiche
```