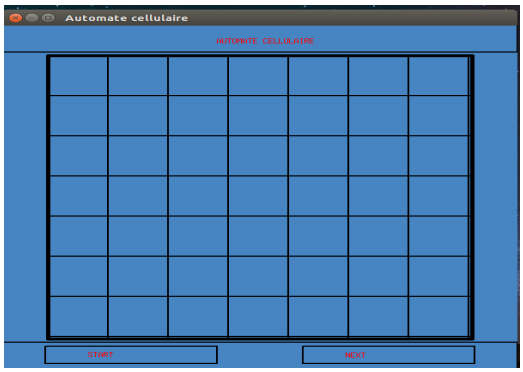
Projet Programmation Fonctionnelle 2015-2016 : Automates Cellulaires

Nom : ABKARI
Prénom : Lydia

Nom : DJEBALI
Prénom : Wissam

Le but de ce projet est de concevoir un automate cellulaire



Le programme permet d'afficher une génération initiale contenue dans le fichier passé en argument ainsi que les générations suivantes .

Les types définies dans notre programme sont contenue dans le fichier "type.ml " :

state: un caractère qui décrit l'état de la cellule, prend la valeur 'A' si la cellule est vivante ou 'D' si elle est morte.

Generation : c'est un tableau à deux dimension de state.

rule : c'est une règle de l'automate, elle correspond à une suite d'état qui décrit l'état de la cellule courante ainsi que l'état de ces voisines nord, est, ouest, sud.

Automaton : Un tableau de règles.

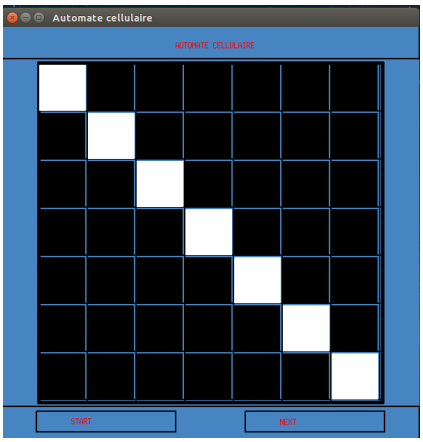
formule = Vrai | Faux
|Var of string
|Neg of formule
|Et of formule * formule
|Ou of formule * formule

les Exceptions :

IncorrectFile : est une exception utilisée lors de la lecture du fichier passé en argument, elle est déclenchée si le fichier n'est pas bien écrit.

SyntaxFile : ex : présence d'une règle mais mal écrite

Le programme se décompose en plusieurs parties :



1. Initialisation :

lire les donnes d'un fichier donné en arguments
et sauvegarde la spécification
de l'automate cellulaire récupéré ainsi
que la génération initiale pour cet automate .

L'ensemble des fonctions implémentées sont :

readf: lit le fichier passé en argument et renvoie une liste de string.

get_dim: renvoie la dimension de la grille.

get_string_line: renvoie la ligne ou se trouve le mot "string " passé en argument, dans la liste.

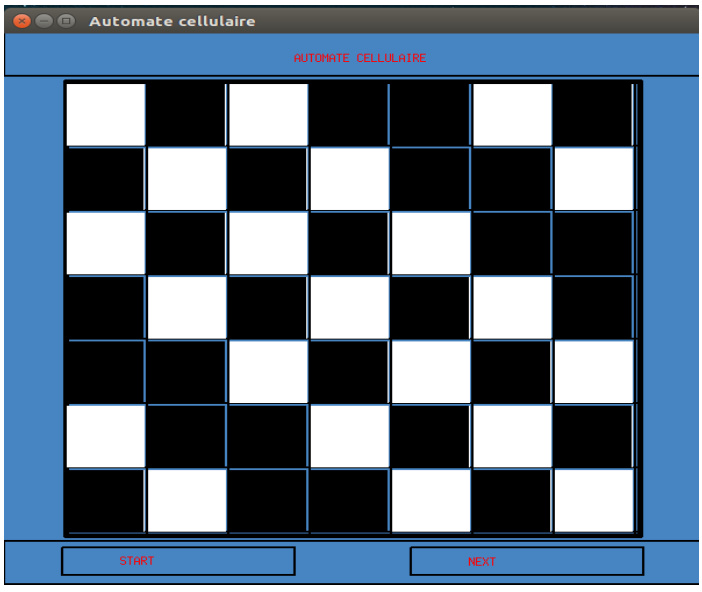
get_lines: prend en argument une liste de string et deux entiers et renvoie le contenue de la liste se trouvant entre les deux lignes dans une liste.

string_to_state_list: prend en argument un string et renvoie une liste d'états. cette fonction permet de récupéré le contenue du fichier sous forme d'une liste.

list_to_generation: transforme le contenue d'une liste en une génération.

list_to_rules: transforme le contenue d'une liste en un automate

parse: prend en argument un fichier et renvoie la dimension de la grille, la génération initiale, ainsi que les règles de l'automate



2. Afficher :

Dans cette partie on va pouvoir
afficher la génération initiale
de notre automate

print: prend en argument un état et l'affiche sur la console.

show_generation: affiche la génération initiale de l'automate

3. Simuler :

Dans cette partie , on calcul pour un automate donné sa génération suivante.
Les différentes fonctions utilisées sont :

right_cell
left_cell
north_cell
south_cell

ces fonctions permettent d'obtenir l'état de chacune des cellules voisine de notre cellule courante.

Voisin : permet de convertir l'état de la cellule courante et celui de ces voisines en une règle.

contains_rule : permet de vérifier si un automate contient une règle donné et renvoie true si c'est oui et false sinon

next_generation : cette fonction permet d'obtenir la génération suivante de notre automate.

4. Modéliser en Calcul Propositionnel :

index : nous permet de récupérer le numéro de la case se trouvant à la linge i et à la colonne c.

indexation : permet de récupérer le numéro de la linge et de la colonne c ou se trouve la case dont le numéro est passé en argument.

right_index :
val north_index :
val south_index :
val left_index :

renvoie les coordonnées des cellules voisines de la cellule à la position (i,j) dans le tableau de génération

neg_literal : traduit un etat en une coordonnee de case
ex:neg_literal Val('A') 10 renvoie Neg (Var(x10))
neg_literal Val('D') 10 renvoie Var(x10)

extract_rules : elle extrait d'un automate toutes les règles telle que la cellule courante est l'état voulu
ex : extract_rules automaton 'A' → renvoie toutes les règles qui se terminent par 'A'.

gen_rules : génère toutes les règles telle que la cellule courante est l'état souhaité.
Ex : gen_rules "A" → génère toute les règles qui se terminent par 'A'

complementaire : renvoie les règles complémentaire d'un automate

tradaux :
ex : si on a une grille de taille 3x3 et des coordonnees de cases (0,0) la fonction va

utiliser la négation sur r ce qui va donner OU(OU(OU(Var(x7),(Var(x3), (Var(x4),Var(x2),(,Neg(Var(x1)))))) .

fus : concaténation de deux listes.

stables

5. Trouver les générations stables :

dimGrid : elle permet à partir de la liste des conjonctions d'obtenir le nombre de case de la grille

tradmin : prend une liste de formule et renvoie une liste de string, on l'utilise pour traduire notre solution obtenue à partir du minisat

create_dimacs : permet de créer le fichier entree.dimacs qu'on donne au minisat.

tradsol : pour traduire la solution du minisat, elle transforme un string en une liste d'entier.
create_gen : permet à partir de la solution renvoyée par le minisat et traduite en entier de créer notre génération stable.

negate_sol : Cette fonction permet d'obtenir la négation de la solution fournit par le minisat qu'on ajoutera par la suite au fichier entree.dimacs afin d'obtenir une nouvelle génération stable
ex : si la solution est « 1 2 3 4 0 » neg renverra « -1 -2 -3 -4 ».

modif_nb_clauses : met à jour le nombre de clauses dans entree.dimacs après modification de celui-ci.

modif_dimacs : récris le fichier entree.dimacs (« même rôle que la fonction creat_dimacs »).

modif_entree : modifie la première ligne et met à jour le fichier entree.dimacs en ajoutant la négation de la solution du minisat

show_stables