1 mai 2014

Projet AF4

UTILISATION DU LOGICIEL:

Le Logiciel est assez simple d'utilisation, il est composé de deux fenêtres qui permettent de gérer deux automates et de les comparer via les différentes méthodes.

Pour comparer les résultats de Moore et Résiduel il faut d'abord les minimiser puis les tester avec « égalité mini ».

Les erreurs ne sont pas gérer mais le logiciel ne plante pas. Ainsi si vous tenter de générer un automate sans avoir rempli les champs rien ne se passe. De même pour l'import d'une expression.

Vous pouvez sauvegarder les automates a tout moment si l'automate découle d'une expression régulière le nom du fichier sera la dite expression, sinon un simple TimeStamp.

GESTION DES MINIMISATIONS ET DE LA GENERATION:

Chaque minimisations créer un objet de la classe correspondantes car chaque action à besoin d'initialiser ces attributs pour remplir ces fonctions.

PROBLÈMES RENCONTRÉS:

Le principal soucis du logiciel est dans le calcul de l'automate minimal via les résiduel: En effet l'algorithme des résiduel calcul les automate minimaux a partir d'une expression régulière il permet donc de vérifier si deux expression régulière décrivent le même langages : (Minimisation / Homomorphie). Or il y a un problème car l'application strict de l'algorithme vu en TD nécessite de reconnaitre deux langages équivalents.

En effet pour savoir si un nouvelle état doit être créer il faut vérifier que le langage résiduel nouvellement calculé ne décrit pas un langage déjà représenté par un

état, or pour certaine expression régulière cela n'est pas si simple, sinon l'algorithme même des résiduel serait inutile.(Simple logique)

Partant de ce constat la bonne méthode serait de créer un algorithme récursif qui vérifie si deux langage sont équivalent par la méthode des résiduel

- Calcul du résiduel d'une expression régulière par rapport a un caractère : renvoi une expression régulière.
- Verification par la même méthode (récursivité) que l'expression régulière n'a pas déjà était rencontré: Existe t'il un état correspondant a un langage homomorphe.

Il est alors facile a comprendre que la clé d'un tel algorithme est dans les condition de sortie de la récursivité a quel moment considère t'on qu'il il est facile de savoir si deux expressions régulière sont assez simple pour être comparé sans difficulté et sans possibilité d'erreur.

Pour revenir au cas du projet vous constaterai qu'une tentative a était effectuer par l'intermédiaire de la méthode simplification() qui permet de trouver des équivalence entre les expression régulière de façon a éviter la création d'état inutile. Cependant il demeure incomplet et de par le caractère infini du nombre d'expression régulière demeura ainsi.

Exemple : L'algorithme n'arrive pas déterminer que *b(ab)*a+e = (ba)* (ou e est le mot vide) Une récursivité simple permettrait de résoudre ce cas ci, mais ce n'est que repoussé le problème.*

En conclusion il est possible de faire un algorithme de minimisation par les résiduel si est seulement si on restreint l'ensemble des expression régulière qu'il est ca-

*pable de gérer. En effet pour gérer n opération (« . », « + », « * ») distinct il faut dans l'absolue 2**n conditions de sortie distinctes (réduction possible) pour couvrir l'ensemble des égalité possible, (réduction des condition de n! à 2**n) cependant il existera une expression a n+1 opérations distinctes tel que l'algorithme ne marche pas.*

Nous nous excusons du caractère plus ou moins hors sujet de cette première parti mais il nous semblait important de souligné le pourquoi de nos difficultés.