IUT Sénart-Fontainebleau, UPEC
MATH 4.2
IUT Info 2A

Florent Madelaine

2022-23

Programmer des automates avec JFLAP

JFLAP [1] est un logiciel libre qui permet de faciliter l'enseignement de modèles théoriques de calcul comme celui des automates. Le J de JFLAP signifie Java.

Le jar est disponible sur le git de ce cours.

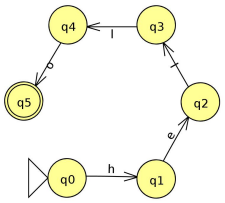
Prise en main : simuler

1. Lancez JFLAP et ouvrez l'automate sauvegardé ici `https://www.dropbox.com/s/kgkfz6146m8947n/TP2NDA.jff`.
2. Allez dans le menu `Input` et choisissez `Step by State`. Entrez le mot babb et observez la simulation des exécutions parallèles étape par étape en appuyant sur `step`.
3. Testez ensuite avec le mot baaa.
4. Relancez les tests ci-dessus. Cette fois essayez les autres options.
   - `Reset` permet de revenir au début du calcul
   - `Trace` après sélection en bas d'une exécution permet de montrer cette exécution depuis le début.
   - `Remove` élimine l'exécution sélectionnée
   - `Freeze`/`Thaw` permet de mettre en pause ou de reprendre une exécution (on perd la synchronisation)
5. Vous pouvez aussi tester sans les étapes avec `Fast Run` voir plusieurs entrées en même temps avec `Multiple Run` (tous les deux dans le menu `Input`). Faites le par exemple pour les deux mots précédents.
6. Pour information, `Step with closure` fait la même chose que `step by State` sauf pour les automates ayant des transitions avec le mot-vide (mais on ne les considère pas dans notre cours).

1. `http://www.jflap.org/`

Hello world

Votre but est de créer l'automate dessiné ci-dessous.



1. Cliquez sur `File` puis `New`.
2. JFLAP permet de gérer différents modèles de calcul. Nous allons nous en tenir aux automates finis. Cliquez sur `Finite Automaton`. Vous avez accès à une nouvelle fenêtre en mode éditeur.
   - Cliquez sur le petit cercle avec un q à l'intérieur, puis sur le canevas pour ajouter des états.
   - Les deux derniers boutons permettent de faire du undo/redo.
   - La flèche fine permet d'ajouter une transition entre 1 ou 2 états (ne pas oublier de mettre une lettre).
   - La flèche plus épaisse à gauche permet de sélectionner, déplacer, éditer les propriétés des différents éléments (par exemple de faire un état initial).
   - *As for the jolly Roger I bet you know what it does.*

Programmer avec JFLAP

Le but principal de ce TP est de programmer avec JFLAP. Comme pour un programme en C, il faut tester régulièrement pour éviter les bugs. Ici c'est plus simple, on ne fait pas de print, on teste avec des mots.

Exercice 1. Dessinez les automates qui reconnaissent :

(pour l'alphabet $\{a,b\}$)

1. les mots de 2 lettres ;
2. les mots contenant exactement 2 a ;
3. les mots contenant 2 a consécutifs ;
4. les mots commençant par b ;
5. les mots se terminant par bab ;
6. le langage réduit au mot vide ;

(pour l'alphabet $\{a,b,c\}$)

7. les mots qui contiennent $aabcc$;
8. les mots de taille multiple de 3 ;
9. les mots de la forme $abcabcabc...$;
10. les mots dont la première lettre est égale à la dernière.

Exercice 2. Construire un automate d'alphabet $\{a,b\}$ correspondant à l'expression régulière

$$a(a+b)^*b$$

Un peu plus technique : calcul modulaire en base 2

L'objectif est de construire des automates qui trient les mots selon le reste modulaire des nombres qu'ils représentent.
On note n un nombre entier et $\overline{n}$ une représentation binaire de ce nombre, avec éventuellement des zéros à gauche. Par exemple, si $n=$ cinq, on peut avoir $\overline{n}=101$ ou bien $\overline{n}=000101$, etc. On a alors $\overline{2n}=\overline{n}0$ et $\overline{2n+1}=\overline{n}1$. Autrement dit, ajouter un 0 à droite d'un nombre écrit en binaire revient à multiplier ce nombre par 2, tandis qu'ajouter un 1 revient à le multiplier par 2 puis ajouter 1 au résultat. Par exemple 1010 est une représentation binaire de dix (deux fois cinq), et 0001011 est une représentation binaire de onze (deux fois cinq plus un).

Exercice 3.
On travaille modulo 3.

1. Complétez le tableau ci-dessous :

n	$2n$	$2n+1$
$\equiv 0 \mod 3$		
$\equiv 1 \mod 3$	$\equiv 2 \mod 3$	
$\equiv 2 \mod 3$		

2. Les automates « modulo 3 » auront trois états correspondant aux trois restes modulo trois possibles. À partir du tableau précédent, en passant aux représentations binaires des entiers, complètez la table de transition ci-dessous :

	0	1
zéro		
un	*deux*	
deux		

3. Construisez un automate déterministe d'alphabet {0,1}, reconnaissant parmi les mots binaires rencontrés, ceux qui représentent des entiers congrus à 1 modulo 3.
4. Utilisez JFLAP pour tester votre design.

Exercice 4. Construisez un automate avec JFLAP qui accepte les multiples de 9 (les nombres sont représentés sur un alphabet binaire).

Exercice 5. Construisez un automate avec JFLAP qui accepte les mots binaires représentant des entiers n tels que :

$$\begin{cases} n \neq 1 \mod 3 \\ n = 2 \mod 5 \end{cases}$$