Feuille de travaux pratiques n^o 5
Automates à états finis

Le logiciel JFLAP utilisé au TP 3 comporte un module très intéressant pour créer graphiquement des automates à états finis, les utiliser pour la reconnaissance du langage rationnel associé et pour effectuer plusieurs transformations comme la transformation en un automate déterministe ou minimal ou en une grammaire formelle.

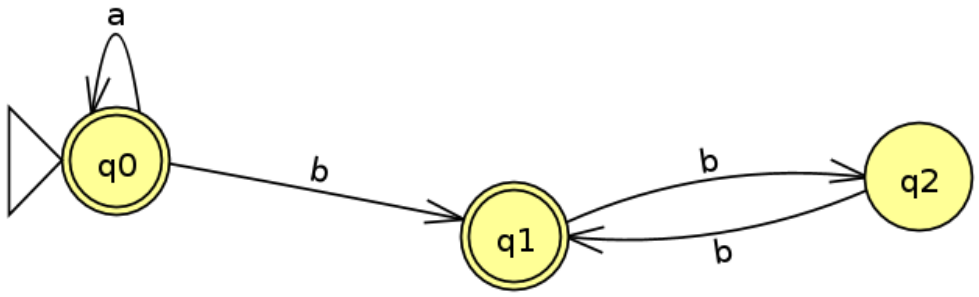
Exercice 5.1 (Tutoriel sur les automates avec JFLAP)

JFLAP est téléchargeable depuis le site `http://www.jflap.org/` après inscription. Une copie est disponible sur Madoc dans la section des TP du module X5I0050 – Langages et Automates.

Pour utiliser le programme, il faut le télécharger et le lancer depuis un terminal en tapant `java -jar JFLAP.jar`. Un menu s'affiche alors permettant de sélectionner un des modules du logiciel.

Le module sur les automates à états finis correspond au menu *Finite Automaton*. On peut ensuite dessiner un automate à l'aide de l'éditeur. Les boutons permettent soit de déplacer un état, de regarder son statut ou de le rendre initial ou final (flèche vers le haut), soit de créer un nouvel état (cercle), soit d'ajouter une transition (flèche vers la droite), soit de supprimer un état ou une transition (tête de mort). Les flèches en arc de cercle permettent de revenir en arrière ou en avant dans l'historique de la construction de l'automate.

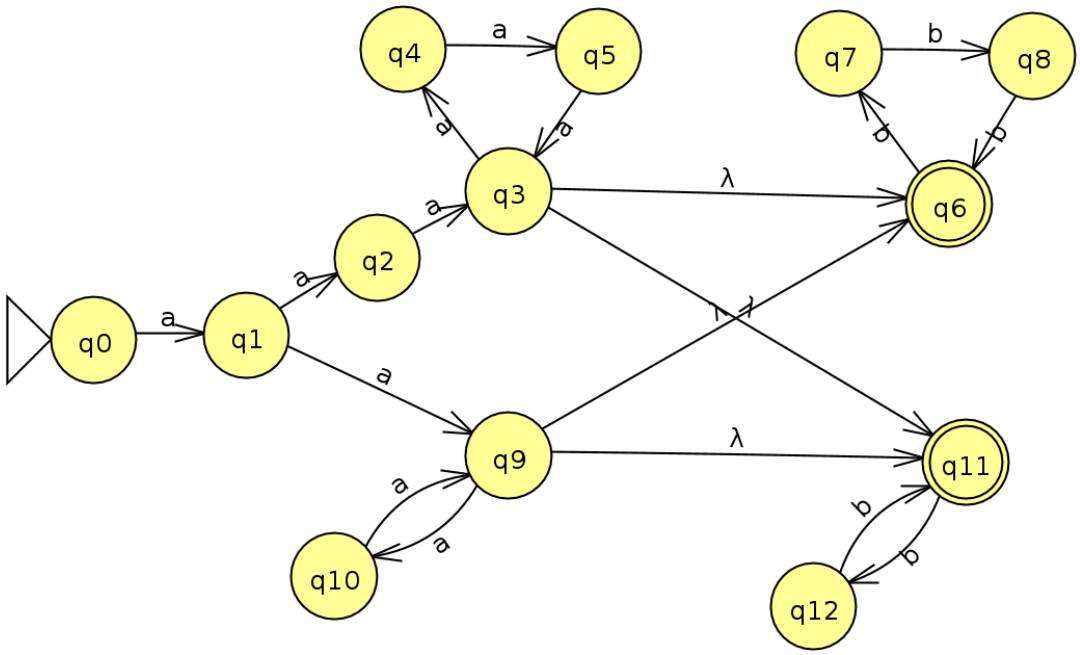
Utiliser l'éditeur pour entrer l'automate suivant :



Nous pouvons ensuite soit simuler l'automate sur une chaîne en entrée, soit vérifier si des chaînes sont acceptées ou non par l'automate. Pour simuler l'automate, il faut cliquer sur *Input* → *Step with Closure...*. On entre ensuite la chaîne à tester, par exemple $aabbb$. On peut ensuite voir les étapes de l'acceptation de l'entrée en cliquant plusieurs fois sur *Step*. Les étapes, aussi appelées configurations, correspondent à un état et à une position sur la chaîne en entrée donnant le préfixe déjà lu par l'automate (en grisé) et la partie qui reste à lire (en gras). Le menu *Input* → *Fast run...* permet de voir toutes les transitions menant à l'acceptation d'une chaîne sous forme d'un diagramme vertical.
Pour tester si un ou plusieurs mots sont acceptés par l'automate, il faut utiliser le menu *Input* → *Multiple Run*. Les mots sont saisis dans la colonne *Input* puis on clique sur *Run Inputs*.
Entrer les chaînes : $aabbb$, abb, $bbbb$, ab et tester leur acceptation. Quelle est l'expression régulière qui correspond à cet automate ?

Exercice 5.2 (Un automate non-déterministe)

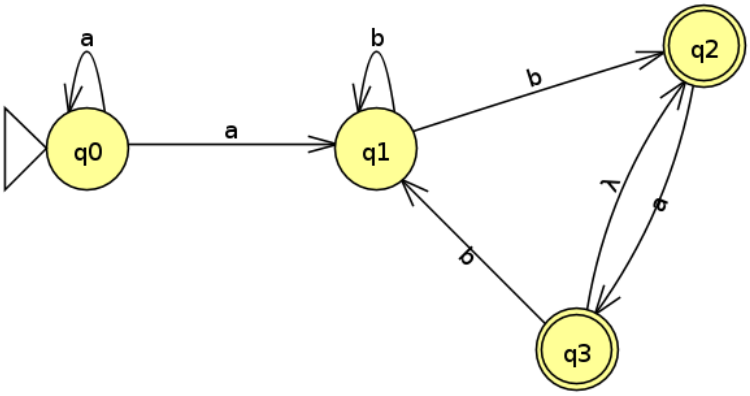
JFLAP permet de saisir des automates non-déterministes soit en utilisant une transition ϵ (notée par JFLAP avec un λ) soit en utilisant des transitions ambiguës (même état de départ, même symbole mais état d'arrivée différent). La création de ce type d'automate n'est pas différente de celle d'un automate déterministe.
Utiliser l'éditeur pour entrer l'automate suivant :



Un automate non-déterministe a plusieurs possibilités à chaque étape de l'acceptation d'une chaîne. On peut voir cela en partant de la chaîne $aaaabb$ dans *Input* → *Step with Closure...* puis en cliquant plusieurs fois sur *Step*.
Comment interpréter la couleur des configurations (bleu, vert ou rouge) ? Tester ensuite les mots suivants (avec *Input* → *Fast run...*) : aa, $aabbb$, a, $aaaaa$ puis donner l'expression rationnelle qui correspond à cet automate.

Exercice 5.3 (Transformation en automate déterministe)

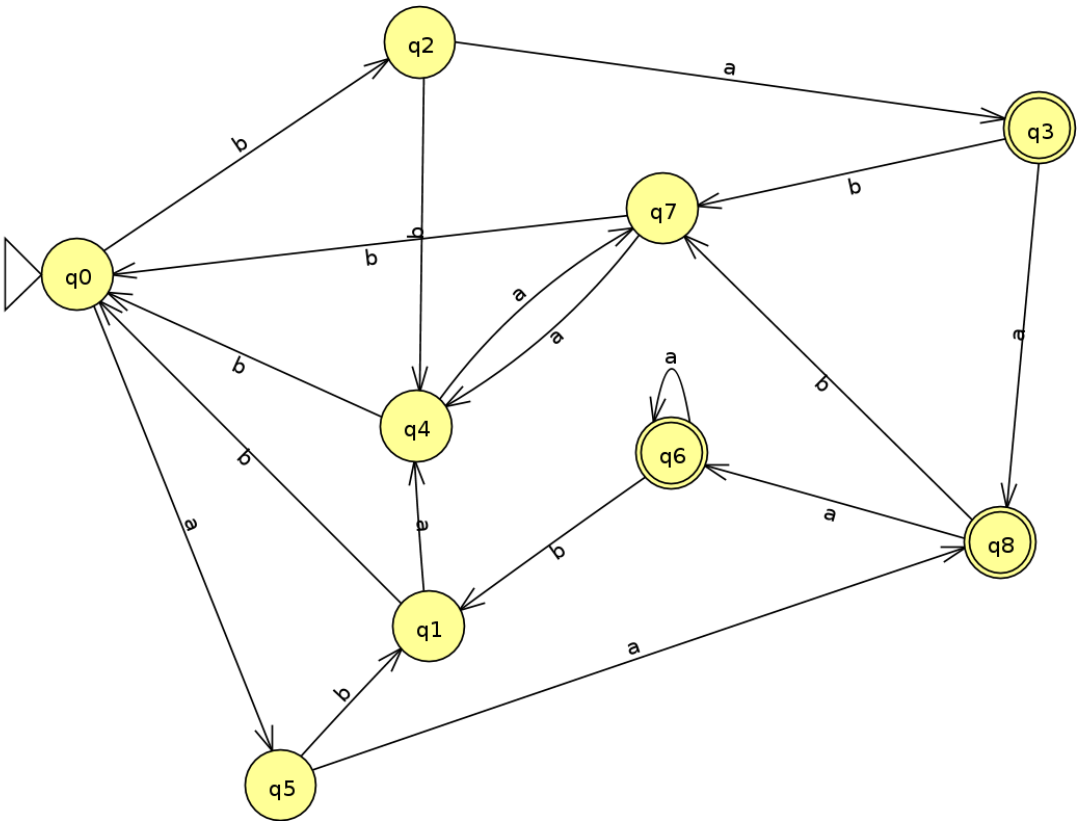
JFLAP implémente un algorithme permettant de transformer un automate non-déterministe en un automate déterministe. Pour comprendre ce mécanisme sur un exemple, entrer l'automate non-déterministe suivant :



Ensuite, on peut utiliser le menu *Convert* → *Convert to DFA* pour la conversion. L'algorithme part de l'état initial puis construit progressivement des états correspondant à des ensembles d'états de l'automate d'origine qui sont atteignables depuis un ou plusieurs autres ensembles d'états déjà calculés. L'interface permet de voir les étapes de la construction des états (bouton *(S)tate Expander* puis on clique sur les états) ou bien de voir directement le résultat final (bouton *Complete*). On peut utiliser ce nouvel automate en cliquant sur *Done*.
Vérifier que les deux automates acceptent ou refusent de la même manière les chaînes aaa, ab, $abbaabb$, $abaab$ et donner l'expression rationnelle correspondante.

Exercice 5.4 (Transformation en automate minimal)

JFLAP implémente un algorithme permettant de transformer un automate déterministe en un automate déterministe minimal. Pour tester cette fonctionnalité, entrer l'automate déterministe suivant :



Ensuite, nous pouvons utiliser le menu *Convert* → *Miniminize DFA* pour la réduction du nombre d'états. L'algorithme essaye de rassembler les états en classes d'équivalence ayant la propriété d'accepter les mêmes chaînes. Pour cela, il part d'une classe constituée de tous les états puis les divise en sous-classes s'il s'aperçoit qu'ils ne sont pas équivalents par rapport à un symbole. Le processus de division des classes s'arrête lorsque les états ne sont plus distinguables dans une classe. Il construit ensuite l'automate minimal sur ces classes d'états qui deviennent les états du nouvel automate en calculant les transitions entre les classes d'états à partir des transitions entre les états de l'automate d'origine.

Pour mettre en pratique cet algorithme en deux phases, on peut demander au logiciel de faire la partition des états étape par étape en cliquant sur une classe et en demandant une partition automatique (bouton *Auto Partition*) ou bien de le faire de manière récursive pour la classe sélectionnée avec le bouton *Complete Subtree*. Ensuite, la seconde phase calcule les transitions et on peut le faire pas à pas (bouton *Hint*) ou bien en une fois (bouton *Complete*).

Calculer l'automate minimal et vérifier sur aa, $babbaa$, ab et bab qu'il donne les mêmes résultats que l'automate initial. Trouver une expression rationnelle pour cet automate.

Exercice 5.5 (Transformation en grammaire formelle)

Le menu *Convert* → *Convert to Grammar* permet de transformer un automate à états finis (déterministe ou non) en une grammaire formelle. Appliquer la méthode à l'automate non-déterministe de l'exercice 4.3. Quel est le type de la grammaire résultat ? Comment pourrait-on obtenir une grammaire rationnelle de type 3 ?