Langages formels (11X003) - Automne 2025

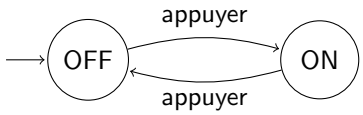
2. Automates finis (déterministes)

Enseignant: Arnaud Casteigts

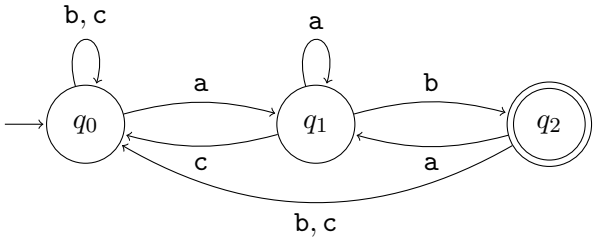
Assistants: A.-Q. Berger & M. Marseloo
Moniteurs: N. Beghdadi & E. Bussod

2.1 Automates finis

Un **automate fini** (AF) est un modèle de machine très rudimentaire, qui représente l'évolution d'un système en fonction d'évènements. Un AF est composé d'un ensemble d'**états** (représentés par des cercles) et de **transitions** qui permettent de changer d'état lorsqu'un évènement se produit (flèches). L'**état initial** est indiqué par une flèche venant de l'extérieur. Voici un exemple :



Cet automate modélise l'état d'un interrupteur, qui bascule entre ON et OFF à chaque appui. L'état initial est en position éteinte (OFF). Nous allons utiliser des automates de ce type pour *analyser des mots*. Voici un exemple d'AF plus complexe, ayant trois états que nous appelerons q_0, q_1, q_2 :



Il s'utilise comme suit : étant donné un mot (extérieur à l'automate), par exemple `acbab` sur l'alphabet $\{a, b, c\}$, nous allons *lire* ce mot caractère par caractère, en empruntant à chaque étape la transition qui correspond au caractère lu. Au départ, nous sommes sur l'état initial q_0. La lecture du premier `a` nous envoit donc sur q_1, puis `c` nous renvoit sur q_0, puis `b` nous laisse sur q_0, `a` nous envoit à nouveau sur q_1, et enfin `b` nous envoit sur q_2. À ce stade, la lecture du mot est terminée. En l'occurrence, nous nous trouvons sur un état spécial (double

cercle) appelé un **état terminal** (ou **final**) de l'automate, nous allons donc **accepter** le mot `acbab`. Si la lecture s'était terminée sur un autre état, nous aurions **rejeté** le mot.

Ainsi, ce type d'automate permet de reconnaître des mots, et par conséquent, des langages ! Quel langage cet automate reconnaît-il ? Réponse : tous les mots qui...

2.1.1 Définition mathématique

Un **automate fini déterministe** (AFD) est un 5-tuple $(Q, \Sigma, \delta, q_0, F)$ où

- Q est un ensemble fini d'états,
- Σ est un alphabet,
- $\delta : Q \times \Sigma \to Q$ est une fonction de transition, qui pour chaque état de Q et chaque symbole de Σ indique dans quel état de Q se déplacer,
- $q_0 \in Q$ est l'état initial,
- $F \subseteq Q$ est l'ensemble des états finaux.

Notez qu'il peut y avoir plusieurs états finaux (F est un *sous-ensemble* de Q), mais un seul état initial (q_0 est un élément de Q). Le comportement de δ est complètement déterministe, dans le sens où pour un état de départ et un symbole donnés, la transition ne peut aller que vers *un seul* état. C'est pour cela que nous qualifions ces automates de *déterministes*. Nous verrons plus tard un type d'automate qualifié de *non déterministe* (AFN). Un AF peut donc être un AFD ou un AFN, mais n'anticipons pas.

Dans l'exemple ci-dessus, on a $Q = \{q_0, q_1, q_2\}, \Sigma = \{a, b, c\}, q_0 = q_0$ (c'est souvent implicite) et $F = \{q_2\}$. On peut représenter δ par une liste de triplets $(q_i, \mathtt{a_i}, q_j)$ signifiant chacun que $\delta(q_i, \mathtt{a_i}) = q_j$, par exemple $(q_0, \mathtt{a}, q_1), (q_0, \mathtt{b}, q_0), \ldots$, ou de manière équivalente, par une table :

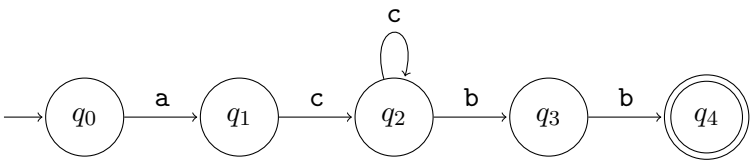
état de départ	symbole	état d'arrivée
q_0	`a`	q_1
q_0	`b`	q_0
q_0	`c`	q_0
$\ldots$	$\ldots$	$\ldots$

Une fonction très utile est $\delta^* : Q \times \Sigma^* \to Q$ qui indique comment l'état évolue en lisant *plusieurs* symboles d'affilée (rappelons que Σ^* correspond à l'ensemble de tous les mots sur Σ). Par exemple, ici, $\delta^*(q_0, \mathtt{ab}) = q_2$, $\delta^*(q_0, \mathtt{bac}) = q_0$ ou encore $\delta^*(q_1, \mathtt{bca}) = q_1$.

Un automate $A = (Q, \Sigma, \delta, q_0, F)$ accepte (reconnaît) un mot w sur Σ si et seulement si $\delta^*(q_0, w) \in F$. Le **langage accepté** (ou **reconnu**) par A est l'ensemble des mots qu'il accepte, à savoir $L(A) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$.

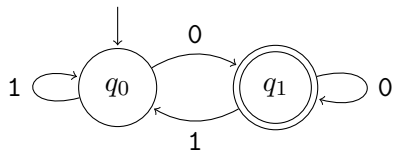
2.1.2 Exemples

Exemple 2.1. Dans le chapitre 1, nous avons évoqué le langage de tous les mots sur l'alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ qui commencent par `a`, suivis d'un ou plusieurs `c`, puis se terminent par `bb`. Ce langage peut être reconnu par l'automate suivant :



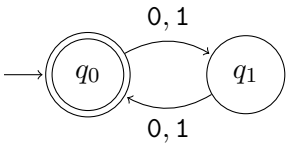
Notez que cet automate ne spécifie pas le comportement à adopter dans tous les cas. Par exemple, il n'indique pas où aller si on lit `a` ou `b` dans l'état q_1. Un tel automate n'est donc pas **complet**. En fait, pour rendre un automate complet, il faut ajouter les transitions non-spécifiées et toutes les diriger vers un même état, appelé **état puits** (à créer), d'où l'on ne ressortira plus et où l'on rejettera ultimement le mot quelle que soit la suite des symboles. Attention, il ne faut pas oublier d'ajouter les transitions partant de l'état puits et allant vers lui-même, afin d'obtenir un automate complet.

Exemple 2.2. Considérons les mots sur l'alphabet $\Sigma = \{\mathtt{0}, \mathtt{1}\}$, en supposant qu'ils représentent des nombres de manière binaire, voici un automate qui reconnaît les *nombres pairs* :



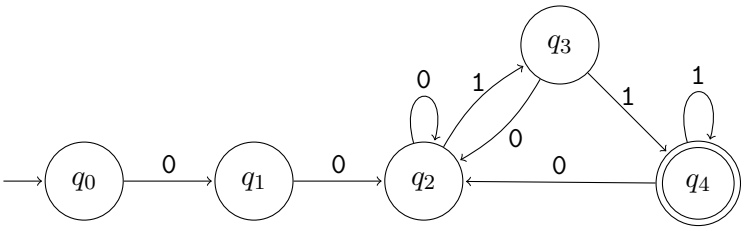
Il est souvent utile, mentalement, d'associer à chaque état une définition textuelle, par exemple ici l'état q_1 correspond à "je viens de lire un 0" et l'état q_0 correspond à "je viens de lire un 1 (ou je n'ai encore rien lu)". Si le mot se termine quand "je viens de lire un 0", c'est gagné.

Exemple 2.3. Voici un autre automate, toujours sur $\Sigma = \{\mathtt{0}, \mathtt{1}\}$, qui reconnaît maintenant les mots de *longueur* paire :



Notez qu'un état initial peut tout à fait être aussi un état final. Cela implique que le mot vide fait partie du langage (et en effet, la longueur du mot ε est zéro, un nombre pair, on veut donc accepter ce mot). Quelle définition textuelle associez-vous à chaque état ?

Exemple 2.4. Voici un exemple plus complexe. Quel langage reconnaît-il ? Réponse : tous les mots sur $\Sigma = \{0, 1\}$ qui ...



2.1.3 Langages réguliers

Comme vous vous en doutez, les AFDs sont des modèles de machine assez limités. Et en effet, il existe des langages qu'ils ne peuvent pas reconnaître, comme par exemple le langage de tous les mots qui sont des palindromes, ou encore tous les mots qui ont au moins autant de 0 que de 1 (réfléchissez-y un peu).

Quels sont les langages reconnaissables par des AFDs ? (Attention, dans cette question, on monte encore d'un niveau, en considérant des ensembles de langages, c'est à dire des ensembles d'ensembles de suites de symboles :-)).

Il s'avère que cette question est très bien comprise aujourd'hui. Les langages reconnaissables par des AFDs (on parle d'**expressivité** des AFDs) sont les **langages réguliers**. Ils correspondent exactement aux langages que l'on peut définir *inductivement*[1] comme suit :

- $\emptyset$ et $\{\varepsilon\}$ sont des langages réguliers,
- Pour tout $\mathtt{s} \in \Sigma$, $\{\mathtt{s}\}$ est un langage régulier,
- Si L est un langage régulier, alors L^* l'est aussi,
- Si L_1 et L_2 sont des langages réguliers, alors $L_1 \cup L_2$ et $L_1 \circ L_2$ le sont aussi.

Cette définition implique aussi que L^+ est régulier si L est régulier, car $L^+ = L \circ L^*$.

Nous verrons plus tard comment on peut montrer cette équivalence, qui n'est pas triviale. Puis nous verrons d'autres modèles dont l'expressivité est plus grande, pour finir sur les *machines de Turing* dont l'expressivité est (supposée être) la même que celle de nos ordinateurs.

1. Inductivement : qui se fait référence à lui-même (on dit aussi récursivement).