TP 3.5 : Extensions des automates finis

Environnement

Nous travaillerons en mode interprété pour ce TP. Nous vous laissons le choix de l'environnement, selon vos habitudes. Vous pouvez utiliser :

- L'interpréteur en ligne BetterOCaml : https://BetterOCaml.ml/
- Emacs, muni du mode Tuareg.

Machines de Moore

Informellement, une machine de Moore peut être vu comme un automate fini déterministe, sans états acceptants, mais où chaque état se voit attribuer une lettre d'un "alphabet de sortie Ω" (en plus des lettres de Σ qui étiquettent les transitions). Chaque fois qu'on arrive dans un état, on "produit" la lettre de sortie associée. Ainsi, en lisant un mot de Σ^* avec la machine, on va produire un mot de Ω^*. On peut donc voir une machine de Moore comme un outil réalisant une fonction partielle $f : \Sigma^* \to \Omega^*$.

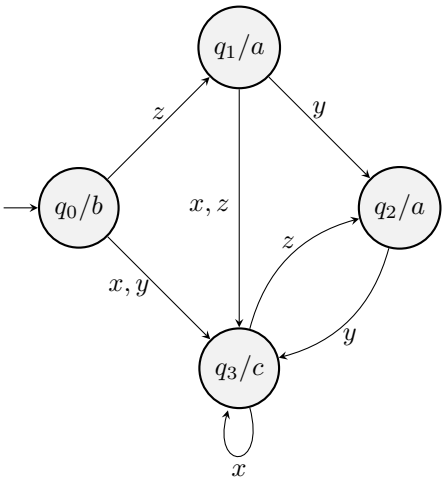
Formellement, une machine de Moore est un sextuplet $(Q, i, \Sigma, \Omega, \delta, \lambda)$ où :

- Q est l'ensemble d'états, et $i \in Q$.
- Σ est l'alphabet d'entrée (sur lequel on lit des mots, comme dans un automate) et Ω est l'alphabet de sortie (celui des mots produits par la machine).
- $\delta : Q \times \Sigma \to Q$ est la fonction de transition qui permet d'avancer dans la machine en lisant des lettres de Σ
- $\lambda : Q \to \Omega$ est la fonction de sortie, qui à chaque état associe la lettre produite en arrivant dessus.

Pour un mot d'entrée $u \in \Sigma^*$ que la machine peut lire, on va suivre un unique chemin $q_0 \xrightarrow{u_1} q_1 \dots \xrightarrow{u_n} q_n$, et le mot de sortie correspondant, noté $f(u)$, s'obtient par la concaténation $\lambda(q_1) \dots \lambda(q_n)$ (la sortie de l'état initial n'est pas prise en compte).

Voici un exemple de Machine de Moore, avec $\Sigma = \{x, y, z\}$ et $\Omega = \{a, b, c\}$. Pour chaque état q, on représente graphiquement l'état avec l'indication $q/\lambda(q)$.

Dans cet exemple, on aurait $f(zyyx) = aacc$; $f(yxz) = cca$, etc.



1. En vous inspirant du type `'a automate` du TP 3, définissez un type `('a,'b) moore` permettant de représenter une machine de Moore.
2. Implémentez la machine de Moore donnée en exemple précédemment. Vous pouvez considérer les lettres commes des caractères, ou vous définir des types énumérés `type sigma = X | Y | Z` et `type omega = A | B | C`

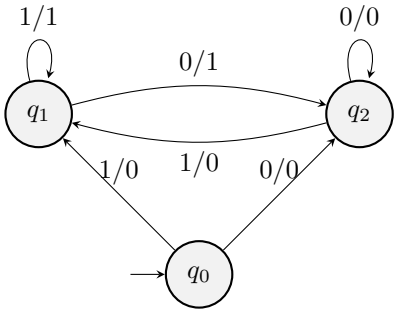
3. Définissez une fonction `lire_lettre : ('a,'b) moore -> int -> 'a -> (int * 'b) option` tel que `lire_lettre m q x` effectue la lecture de la lettre `x` dans l'état `q` et renvoie la paire `(q',b)` représentant l'état d'arrivée et sa sortie, si la transition est possible.
4. Définissez une fonction `lire_mot : ('a,'b) moore -> 'a mot -> ('b mot) option`, qui effectue la lecture d'un mot de type `'a` depuis l'unique état initial d'une machine de Moore, et renvoie le mot de type `'b` correspondant à la sortie.
5. On pose $\Sigma = \Omega = \{a, b\}$. Donnez une machine de Moore qui, pour chaque mot u, produit le mot obtenu en remplaçant chaque occurrence de a par un b et réciproquement. Implémentez-la et testez-la.
6. On pose $\Sigma = \{a, b\}$ et $\Omega = \{b, x, y\}$. Donnez une machine de Moore qui, pour chaque mot u, produit le mot obtenu en remplaçant les occurrences d'ordre impair de a (le 1er a du mot, le 3ème a, le 5ème, etc... par des x, et les occurrences d'ordre pair de a par des y. Implémentez-la et testez-la.

Machines de Mealy

Formellement, une machine de Mealy se définit exactement comme une Machine de Moore, mais on considère cette fois λ comme une fonction de $Q \times A$ dans B, avec la sémantique suivante : quand on part de q en lisant une lettre a, on produit la sortie $\lambda(q, a)$.

Pour un mot d'entrée $u \in \Sigma^*$ que la machine peut lire, on va suivre un unique chemin $q_0 \xrightarrow{u_1} q_1 \dots \xrightarrow{u_n} q_n$, et le mot de sortie correspondant, noté $f(u)$, s'obtient par la concaténation $\lambda(q_0, u_1) \dots \lambda(q_{n-1}, u_n)$.

Voici un exemple de Machine de Mealy, avec $\Sigma = \Omega = \{0, 1\}$. Pour représenter les sorties, comme elles dépendent de l'état et de la lettre qu'on a lu pour y rentrer, on les représentera plutôt sur les transitions, sous la forme x/a, avec la sémantique "en prenant cette transition, on doit lire la lettre x, et on produit la sortie a".



1. Définissez un type `('a,'b) moore` modélisant les machines de Mealy.
2. Implémentez une fonction permettant de lire un mot dans une machine de Mealy, et produisant la sortie correspondante.
3. Que fait la machine de Mealy ci-dessus ? Testez la éventuellement en l'implémentant et en lui faisant lire des mots.
4. Proposez une machine de Mealy qui réalise la fonction $f : \{0,1\}^* \to \{0,1\}^*$ suivante : pour tout $u = u_1 \dots u_n$, $f(u) = v_1 \dots v_n$ tel que pour tout i, v_{i+1} s'obtient par le "ou exclusif" de u_i et de u_{i+1}. (On aura toujours $v_1 = 0$ par convention)

Annexes

Tables de hachage

Pour manipuler des tables d'association (dictionnaires):

- `Hashtbl.create n` Renvoie une table de hachage de taille n.
- `Hashtbl.add table cle valeur` Ajoute l'association clé → valeur à la table de hachage.
- `Hashtbl.replace table cle valeur` Remplace la valeur courante associée à `cle` dans la table par `valeur`.
- `Hashtbl.mem table cle` renvoie `true` ssi la cle a une valeur associée dans la table.
- `Hashtbl.find table cle` renvoie la valeur associée à la clé dans la table.
- `Hashtbl.find_opt table cle` renvoie une option sur la valeur associée à la clé dans la table (c'est-à-dire `None` s'il n'en existe pas, et `Some v` si la valeur `v` a été trouvée.