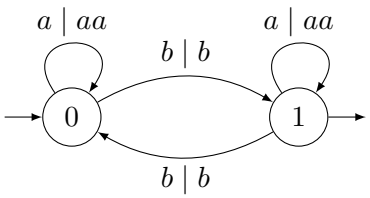
Ambigüité et transducteurs

Dans ce sujet, un **automate** est un quintuplet $\mathcal{A} = \langle Q, A, E, I, T\rangle$ avec Q un ensemble fini et non vide d'états, A un alphabet fini et non vide, $I \subseteq Q$ l'ensemble des états initiaux, $T \subseteq Q$ l'ensemble des états terminaux et $E \subseteq Q \times A \times Q$ l'ensemble des transitions. La transition $(p, a, q) \in E$ est notée par la flèche $p \xrightarrow{a} q$. Un **calcul** c de $\mathcal{A}$ est une séquence finie de transitions qui forment un chemin dans le graphe de $\mathcal{A}$, $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots \xrightarrow{a_n} p_n$: on note un tel calcul $c = p_0 \xrightarrow{a_1 a_2 \cdots a_n} p_n$. L'étiquette de c est le mot $a_1 a_2 \cdots a_n$ de A^*. Le calcul c est acceptant si $p_0 \in I$ et $p_n \in T$.

Le **langage** de $\mathcal{A}$ est le sous-ensemble $\mathcal{L}(\mathcal{A})$ de A^* qui contient l'ensemble des étiquettes des calculs acceptants de $\mathcal{A}$. L'automate $\mathcal{A}$ est dit **déterministe** si I possède exactement un état et pour tout $p \in Q$ et $a \in A$, il existe au plus un état $q \in Q$ tel que $(p, a, q) \in T$. Un état $q \in Q$ est dit **accessible** s'il existe un mot $u \in A^*$ et un calcul $p_0 \xrightarrow{u} q$ avec $p_0 \in I$, et **co-accessible** s'il existe un mot $u \in A^*$ et un calcul $q \xrightarrow{u} p$ avec $p \in T$. Un automate $\mathcal{A}$ est dit **émondé** s'il ne contient que des états qui sont accessibles et co-accessibles.

Un **transducteur** $\mathcal{T}$ est alors un automate sur un alphabet qui est un sous-ensemble fini de $A \times B^*$, avec A et B deux alphabets finis : il doit de plus vérifier la propriété que pour tous états p et q et toute lettre $a \in A$, il existe au plus un mot $u \in B^*$ tel qu'il existe la transition $p \xrightarrow{(a,u)} q$ dans le transducteur. Pour une transition $p \xrightarrow{(a,u)} q$ de $\mathcal{T}$, avec $a \in A$ et $u \in B^*$ (u peut donc être le mot vide ε), on dit que u est la production de la transition. Un calcul est désormais de la forme $p_0 \xrightarrow{(a_1,u_1)\cdots(a_n,u_n)} p_n$. Cette fois, on dit que $a_1 \cdots a_n$ est son étiquette et que $u_1 \cdots u_n$ est sa production. Par abus de notation, on le note souvent $p_0 \xrightarrow{a_1 \cdots a_n | u_1 \cdots u_n} p_n$. La sémantique de $\mathcal{T}$ est désormais une relation $\mathcal{R}(\mathcal{T})$ de A^* dans B^* : elle relie un mot $a_1 \cdots a_n \in A^*$ à tout mot $u_1 \cdots u_n \in B^*$ tel qu'il existe un calcul acceptant d'étiquette $a_1 \cdots a_n$ et de production $u_1 \cdots u_n$. De tout transducteur $\mathcal{T}$, on peut extraire son automate sous-jacent $\mathcal{A}$, qui consiste simplement à supprimer tous les mots de B^* des transitions de $\mathcal{T}$. Le langage de $\mathcal{A}$ est alors le domaine de la relation $\mathcal{R}(\mathcal{T})$.

On représente de manière classique les transducteurs comme un graphe, dont les sommets sont les états et les arcs les transitions : une transition $(p, (a, u), q)$ est représentée par un arc d'étiquette $\ll a \mid u \gg$. Un exemple de transducteur sur les alphabets $A = B = \{a, b\}$ est représenté ci-dessous, où 0 est un état initial et 1 un état terminal.



1. Décrire la relation définie par le transducteur ci-dessus.

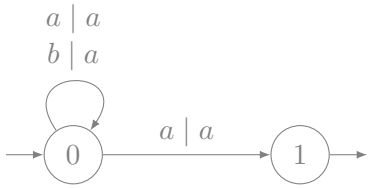
Le domaine de $\mathcal{R}(\mathcal{T})$ est l'ensemble des mots avec un nombre impair de b : de plus, $\mathcal{R}(\mathcal{T})$ relie un tel mot $a^{n_1}ba^{n_2}b\cdots a^{n_p}ba^{n_{p+1}}$ (avec p impair) avec le mot $a^{2n_1}ba^{2n_2}b\cdots a^{2n_p}ba^{2n_{p+1}}$.

2. Considérons à nouveau les alphabets $A = B = \{a, b\}$. Donner des transducteurs représentant les relations R_0 et R_1 suivantes

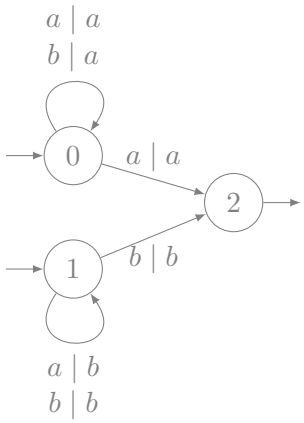
$$R_0 = \{(u, a^{|u|}) \mid u \in A^*, u \text{ se termine par la lettre } a\}$$
$$R_1 = \{(u, b^{|u|}) \mid u \in A^*, u \text{ se termine par la lettre } b\} \cup R_0$$
$$R_2 = \{(u, v) \mid u \in A^*, v \text{ est un sous-mot de } u\}$$

où $|u|$ représente la longueur du mot u, et un sous-mot de $u = a_1a_2\cdots a_n$ est un mot de la forme $a_{i_1}a_{i_2}\cdots a_{i_p}$ avec $1 \leq i_1 < i_2 < \cdots < i_p \leq n$.

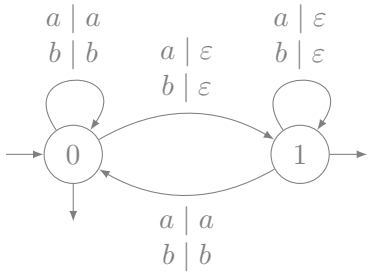
transducteur pour R_0



transducteur pour R_1



transducteur pour R_2



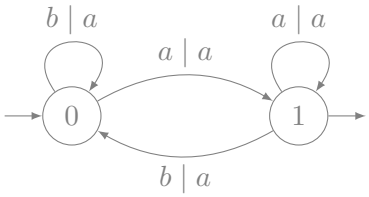
Notons que pour R_2, la solution plus simple consistant à n'avoir qu'un seul état n'est pas formellement correct à cause de la restriction imposée « pour tous états p et q et toute lettre $a \in A$, il existe au plus un mot $u \in B^*$ tel qu'il existe la transition $p \xrightarrow{(a,u)} q$ dans le transducteur » : cela permet d'assurer qu'il n'existe pas plusieurs flèches de p à q avec la même étiquette a, sans quoi l'automate sous-jacent ne serait pas bien défini.

Un transducteur $\mathcal{T}$ est dit **fonctionnel** si la relation $\mathcal{R}(\mathcal{T})$ est en fait une fonction partielle, c'est-à-dire que tout mot de A^* est relié dans R à au plus un mot de B^*. Le transducteur $\mathcal{T}$ est dit **séquentiel** si son automate sous-jacent est déterministe.

3. Laquelle ou lesquelles des relations R_0, R_1 et R_2 sont des fonctions ? Donner un transducteur séquentiel les reconnaissant si c'est possible. Sinon, expliquer pourquoi ce n'est pas possible.

Les relations R_0 et R_1 sont fonctionnelles, mais pas la relation R_2 puisque cette dernière relie le mot a aux mots de $\{\varepsilon, a\}$.

On peut reconnaitre R_0 avec le transducteur séquentiel suivant :



On ne peut pas reconnaitre la relation R_2 avec un transducteur séquentiel, puisque cela impliquerait aisément que la relation est fonctionnelle.

Finalement, on peut montrer que la relation R_1 ne peut pas être reconnue par un transducteur séquentiel : intuitivement, il faudrait connaitre la dernière lettre dès le début pour pouvoir faire le bon choix, on ne sait donc pas quoi produire lors de la première transition. Plus formellement, supposons par l'absurde qu'il existe un transducteur déterministe $\mathcal{T}$ qui reconnait $\mathcal{R}(\mathcal{T}) = R_1$. Depuis l'unique état initial q_i, l'unique transition étiquetée par a doit mener à un état acceptant, puisque le mot a est dans le domaine de R_1, et la transition doit donc produire la lettre a. Mais alors, le mot ab ne sera pas associé à la bonne production, puisqu'il ne faut pas produire la lettre a.

Un automate $\mathcal{A}$ est dit non ambigu si pour tout mot $u \in \mathcal{L}(\mathcal{A})$ du langage de $\mathcal{A}$, il existe exactement un calcul acceptant dont u est l'étiquette. Dans le cas contraire, $\mathcal{A}$ est dit ambigu.

4. Montrer que si $\mathcal{T}$ est un transducteur non fonctionnel, alors son automate sous-jacent $\mathcal{A}$ est ambigu. *[Indication : on pourra utiliser une preuve par contraposée.]* La réciproque est-elle vraie ?

On prouve la contraposée, c'est-à-dire si l'automate sous-jacent $\mathcal{A}$ est non ambigu, alors $\mathcal{T}$ est un transducteur fonctionnel. Supposons $\mathcal{A}$ non ambigu et $\mathcal{T}$ non fonctionnel. Alors, on sait qu'il existe un mot $u \in A^*$ relié par $\mathcal{R}(\mathcal{T})$ à deux mots v et w de B^*. En particulier, cela implique qu'il existe deux calculs acceptants d'étiquette u dans $\mathcal{T}$. Ils se transfèrent en deux calculs acceptants différents d'étiquette u dans l'automate sous-jacent $\mathcal{A}$, ce qui contredit la non ambigüité de $\mathcal{A}$.

La réciproque est fausse. L'automate $\mathcal{A}$ peut être ambigu mais que les calculs sur une étiquette donnée produisent tous le même mot. Comme contre-exemple, on peut prendre deux copies indépendantes d'un même transducteur déterministe : il reste fonctionnel, mais l'automate sous-jacent admet deux calculs acceptants pour chaque mot de son langage.

5. Montrer qu'on peut décider en temps polynomial si un automate donné est non ambigu. *[Indication : on pourra utiliser un automate $\mathcal{A} \times \mathcal{A}$ obtenu en prenant le produit cartésien*

$Q \times Q$ comme ensemble d'états et toujours A comme alphabet, et montrer que $\mathcal{A}$ est non ambigu si et seulement si les états accessibles et co-accessibles de $\mathcal{A} \times \mathcal{A}$ sont de la forme (q, q).]

On note $\mathcal{A} = \langle Q, A, E, I, T\rangle$ et on définit $\mathcal{A} \times \mathcal{A} = \langle Q \times Q, A, E', I \times I, T \times T\rangle$, avec $E' = \{((p, p'), a, (q, q')) \mid a \in A, (p, a, q) \in E, (p', a, q') \in E\}$. Montrons que $\mathcal{A}$ est non ambigu si et seulement si les seuls états accessibles de $\mathcal{A} \times \mathcal{A}$ depuis $I \times I$ sont de la forme (q, q). En effet, $\mathcal{A}$ est ambigu si et seulement s'il existe deux calculs acceptants distincts $c = p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots \xrightarrow{a_n} p_n$ et $c' = p'_0 \xrightarrow{a_1} p'_1 \xrightarrow{a_2} p'_2 \cdots \xrightarrow{a_n} p'_n$, c'est-à-dire si et seulement s'il existe un calcul acceptant $(p_0, p'_0) \xrightarrow{a_1} (p_1, p'_1) \xrightarrow{a_2} (p_2, p'_2) \cdots \xrightarrow{a_n} (p_n, p'_n)$ de $\mathcal{A} \times \mathcal{A}$ dans lequel $p_i \neq p'_i$ pour au moins un indice i. Cela veut donc dire que (p_i, p'_i) est un état accessible et co-accessible.

On peut décider de la non ambigüité de $\mathcal{A}$ en construisant la partie accessible de $\mathcal{A} \times \mathcal{A}$, puis en l'émondant grâce à un parcours en profondeur (en suivant les transitions dans le sens inverse) depuis les états de $T \times T$. On vérifie alors que tous les états restants sont de la forme (q, q). C'est donc un algorithme quadratique en le nombre d'états de $\mathcal{A}$.

6. On cherche désormais à décider la fonctionnalité d'un transducteur $\mathcal{T}$. Pour deux mots $u, v \in B^*$, on note $\mathsf{delai}(u, v)$ la paire (u', v') telle que : soit $u = vu'$ et $v' = \varepsilon$ (si v est un préfixe de u), soit $v = uv'$ et $u' = \varepsilon$ (si u est un préfixe de v). Ainsi, $\mathsf{delai}(u, v)$ n'est pas défini pour tous les mots u et v : par exemple, $\mathsf{delai}(a, b)$ n'est pas défini.

a) Pour quelles paires de mots (u, v) a-t-on $\mathsf{delai}(u, v) = (\varepsilon, \varepsilon)$?

Si $\mathsf{delai}(u, v) = (\varepsilon, \varepsilon)$, c'est que v est un préfixe de u, auquel cas $u = v\varepsilon$, ou bien u est un préfixe de v, auquel cas $v = u\varepsilon$. Dans les deux cas, $u = v$. Réciproquement, on a bien $\mathsf{delai}(u, u) = (\varepsilon, \varepsilon)$.

b) Soient $u, v, w, z \in B^*$. Montrer qu'on peut calculer $\mathsf{delai}(uw, vz)$ à partir de $\mathsf{delai}(u, v)$, w et z.

Si $\mathsf{delai}(u, v)$ n'est pas défini, c'est que u et v ne sont pas préfixes l'un de l'autre. Il en est donc de même pour uw et vz. Sinon, il y a deux cas. Si $\mathsf{delai}(u, v) = (u', \varepsilon)$, alors $u = vu'$, donc $uw = vu'w$. Si z est un préfixe de $u'w$, alors $\mathsf{delai}(uw, vz) = (u'', \varepsilon)$, avec $u'w = zu''$. Si $u'w$ est un préfixe de z, alors $\mathsf{delai}(uw, vz) = (\varepsilon, z')$ avec $z = u'wz'$. Sinon, on a $z = \alpha a\beta$ et $u'w = \alpha b\gamma$, avec a et b deux lettres différentes de B. On a donc $uw = vu'w = v\alpha b\gamma$ et $vz = v\alpha a\beta$, de sorte que uw et vz n'est pas préfixe l'un de l'autre : $\mathsf{delai}(uw, vz)$ n'est donc pas défini dans ce cas. Un raisonnement similaire permet de trouver $\mathsf{delai}(uw, vz)$ dans le cas où $\mathsf{delai}(u, v) = (\varepsilon, v')$.

c) Dans la définition de transducteur, on peut remarquer que l'ensemble B^* peut être remplacé par $B^* \times B^*$, auquel cas le transducteur relie des mots sur l'alphabet A à des paires de mots sur l'alphabet B. Construire un tel transducteur $\mathcal{T} \times \mathcal{T}$ inspiré par l'automate construit dans la question **5**. On suppose l'automate sous-jacent de $\mathcal{T} \times \mathcal{T}$ émondé (c'est-à-dire qu'on supprime les états qui ne sont pas accessibles et co-accessibles). Montrer que $\mathcal{T}$ est fonctionnel si et seulement si les deux propriétés suivantes sont vérifiées :

(i) pour tous calculs $(p_0, p'_0) \xrightarrow{\alpha|(u,u')} (p, p')$ et $(p_1, p'_1) \xrightarrow{\beta|(v,v')} (p, p')$ dans $\mathcal{T} \times \mathcal{T}$, on a $\mathsf{delai}(u, u') = \mathsf{delai}(v, v')$: cela implique qu'on peut définir de manière unique le délai $\mathsf{delai}(p, p')$ entre deux états p et p' ;

(ii) pour tout état terminal (p, p') de $\mathcal{T} \times \mathcal{T}$, $\mathsf{delai}(p, p') = (\varepsilon, \varepsilon)$.

On pose $\mathcal{T}\times\mathcal{T} = \langle Q\times Q, A\times(B^*\times B^*), E', I\times I, T\times T\rangle$ avec $E' = \{((p,p'),(a,(u,u')),(q,q')) \mid a \in A, u, u' \in B^*, (p,(a,u),q) \in E, (p',(a,u'),q') \in E\}$. Grâce à la question **4**, on peut supposer que l'automate sous-jacent de $\mathcal{T}$ est ambigu, sans quoi le transducteur $\mathcal{T}$ est fonctionnel immédiatement.

Montrons dans un premier temps que les deux conditions sont suffisantes. On suppose donc que (i) et (ii) sont vérifiées et on prouve que $\mathcal{T}$ est fonctionnel. On considère donc deux calculs acceptants de $\mathcal{T}$ lisant la même étiquette : $p_0 \xrightarrow{a_1|u_1} p_1 \xrightarrow{a_2|u_2} p_2 \cdots \xrightarrow{a_n|u_n} p_n$ et $p'_0 \xrightarrow{a_1|v_1} p'_1 \xrightarrow{a_2|v_2} p'_2 \cdots \xrightarrow{a_n|v_n} p'_n$. Puisque p_0 et p'_0 sont initiaux, on a $(p_0, p'_0) \in I \times I$ et la propriété (i) implique que $\mathsf{delai}(p_0, p'_0) = (\varepsilon, \varepsilon)$. On a alors $\mathsf{delai}(p_1, p'_1) = \mathsf{delai}(u_1, v_1)$, par la question **6**.b, et plus généralement, par induction, $\mathsf{delai}(p_n, p'_n) = \mathsf{delai}(u_1 \cdots u_n, v_1 \cdots v_n)$. Comme p_n et p'_n sont tous deux terminaux, on a $(p_n, p'_n) \in T \times T$ et la propriété (ii), on sait que $\mathsf{delai}(p_n, p'_n) = (\varepsilon, \varepsilon)$. Ainsi $\mathsf{delai}(u_1 \cdots u_n, v_1 \cdots v_n) = (\varepsilon, \varepsilon)$, ce qui implique, par la question **6**.a, que $u_1 \cdots u_n = v_1 \cdots v_n$. Le transducteur $\mathcal{T}$ est donc fonctionnel.

Réciproquement, supposons que $\mathcal{T}$ est fonctionnel. Supposons d'abord que la propriété (i) ne soit pas satisfaite. Puisque l'automate sous-jacent de $\mathcal{T} \times \mathcal{T}$ est émondé, il existe donc deux calculs acceptants $(p_0, p'_0) \xrightarrow{\alpha|(u,u')} (p, p') \xrightarrow{\gamma|(w,w')} (q, q')$ et $(p_1, p'_1) \xrightarrow{\beta|(v,v')} (p, p') \xrightarrow{\gamma|(w,w')} (q, q')$ avec $\mathsf{delai}(u, u') \neq \mathsf{delai}(v, v')$. Or $uw = u'w'$ et $vw = v'w'$ puisque $\mathcal{T}$ est fonctionnel. Ces deux équations impliquent que u ou u' sont préfixes l'un de l'autre, et de même pour v et v' : ainsi, $\mathsf{delai}(u, u')$ et $\mathsf{delai}(v, v')$ sont bien tous les deux définis. Sans perte de généralité, supposons que $\mathsf{delai}(u, u') = (u'', \varepsilon)$ de sorte que $u = u'u''$. L'équation $uw = u'w'$ se réécrit donc en $u''w = w'$. On déduit alors de $vw = v'w'$ que $vw = v'u''w$ et donc que $v = v'u''$. On a donc $\mathsf{delai}(v, v') = (u'', \varepsilon)$ ce qui contredit le fait que $\mathsf{delai}(u, u') \neq \mathsf{delai}(v, v')$.

La propriété (i) est donc nécessairement satisfaite. Supposons finalement que la propriété (ii) ne soit pas satisfaite. Alors il existe une paire d'état terminaux (p, p') de délai $\mathsf{delai}(p, p') \neq (\varepsilon, \varepsilon)$. Puisque l'automate sous-jacent de $\mathcal{T} \times \mathcal{T}$ est émondé, on sait qu'il existe un calcul acceptant $(p_0, p'_0) \xrightarrow{\alpha|(u,u')} (p, p')$. Par la propriété (1), on sait que $\mathsf{delai}(u, u') = \mathsf{delai}(p, p')$. La question **6**.a assure alors que $u \neq u'$, ce qui contredit la fonctionnalité de $\mathcal{T}$.

d) En déduire un algorithme en temps polynomial pour décider la fonctionnalité d'un transducteur $\mathcal{T}$. Détailler sa complexité.

On construit le transducteur $\mathcal{T} \times \mathcal{T}$ et on l'émonde. Ensuite, on cherche à étiqueter tous les états (p, p') par son délai grâce à un parcours en profondeur de l'automate sous-jacent. Lorsqu'on veut visiter un état (p, p'), s'il n'a pas encore été visité précédemment, on lui donne un délai grâce à la question **6**.b. S'il a été précédemment visité mais que son délai est le même que celui qu'on obtiendrait grâce à la question **6**.b, on continue. Dans le cas contraire, on arrête

l'exploration et on conclut que $\mathcal{T}$ n'est pas fonctionnel. À la fin, on regarde si tous les états terminaux ont un délai $(\varepsilon, \varepsilon)$, auquel cas on conclut que $\mathcal{T}$ est fonctionnel.

Notons n le nombre d'états de $\mathcal{T}$, m son nombre de transitions et K la taille maximale d'une production sur une transition. Le nombre de transition de $\mathcal{T} \times \mathcal{T}$ est m^2 et on peut le construire avec une complexité $O(m^2)$. On peut également l'émonder avec cette même complexité. Le calcul du délai pour un état est proportionnel à la longueur des mots du délai, eux-mêmes bornés par Kn^2, puisqu'il suffit de considérer des chemins sans boucles pour calculer le délai d'une paire d'états (et que les délais ne sont pas plus longs que les productions des calculs). Au total, on peut donc faire le parcours en profondeur pour calculer les délais avec une complexité $O(Kn^2m^2)$. Le test final sur les états terminaux a une complexité $O(n^2)$.

Tiré de *Squaring transducers : an efficient procedure for deciding functionality and sequentiality*, par Marie-Pierre Béal, Olivier Carton, Christophe Prieur et Jacques Sakarovitch, Theoretical Computer Science 292 (2003).