DOMAINE : Non olympique
NIVEAU : Intermédiaire
CONTENU : Cours

AUTEUR : Vincent JUGÉ
STAGE : Montpellier 2013

Automates

Ce cours a vocation à présenter aux élèves le concept d'automates finis, ainsi que le concept voisin de transducteurs finis. Un automate peut être vu comme une machine, ou bien un programme informatique, qui représente un ensemble : ce programme décide si, oui ou non, un mot donné appartient à l'ensemble en question. De même, un transducteur peut être vu comme un programme informatique, qui représente une fonction transformant les mots en d'autres mots : le programme permet de calculer explicitement cette fonction.

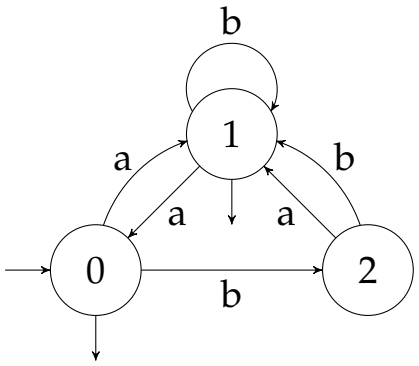
- Automates -

Qu'est-ce qu'un automate ? Alors qu'il existe bien des types d'automates (permettant de travailler sur mots ou bien des arbres, qu'ils soient finis ou infinis), nous nous concentrerons ici sur un type d'automates bien particulier : les automates traitant de **mots finis**.

En clair, comment tout cela fonctionne-t-il ? Tout d'abord, on décide de se fixer un alphabet (ensemble de lettres) fini $\mathcal{A}$. Un **mot fini** est une suite finie d'éléments de $\mathcal{A}$, et c'est sur de tels mots que l'on va travailler ; on note couramment $\mathcal{A}^*$ l'ensemble de ces mots finis dont les lettres appartiennent à $\mathcal{A}$. Un automate peut être vu comme un graphe orienté fini $\mathcal{G}$, un peut spécial :

- il admet une ou plusieurs arêtes **entrantes**, qui viennent de nulle part et arrivent en un sommet de $\mathcal{G}$;
- il admet des arêtes **sortantes**, qui partent d'un sommet de $\mathcal{G}$ et n'aboutissent nulle part ;
- il admet des arêtes **normales**, permettant d'aller d'un sommet de $\mathcal{G}$ à un autre ; de surcroît, chacune de ces arêtes normales est étiquetée par une lettre, c'est-à-dire un élément de $\mathcal{A}$.

Voici un exemple d'automate $\mathcal{G}$ pour l'alphabet $\mathcal{A} = \{a, b\}$:



Automate $\mathcal{G}$

Le graphe $\mathcal{G}$ admet une arête entrante (qui arrive au sommet 0), deux arêtes sortantes (qui partent des sommets 0 et 1) et six arêtes normales. Notons d'ailleurs que le graphe $\mathcal{G}$, s'il est orienté, n'est pas nécessairement **simple** : il peut tout à fait contenir des boucles (l'arête b qui relie le sommet 1 à lui-même) et des arêtes multiples (les arêtes a et b qui vont du sommet 2 au sommet 1).

Cependant, on a dit plus haut qu'un tel graphe pouvait être vu comme un programme informatique, voire comme un **ensemble** de mots : il nous faut donc décrire maintenant pourquoi. Considérons un mot fini $\mathbf{u} = u_1 u_2 \dots u_n$, de longueur n. Un **chemin acceptant** pour le mot $\mathbf{u}$ sera une suite de sommets $s_0, s_1, \dots, s_n$ (de longueur $n+1$) telle que

- il existe une arête entrante qui arrive au sommet s_0 ;
- pour tout entier $k \in \{1, 2, \dots, n\}$, il existe une arête normale, étiquetée par la lettre u_k, qui part du sommet s_{k-1} et arrive au sommet s_k ;
- il existe une arête sortante qui part du sommet s_n.

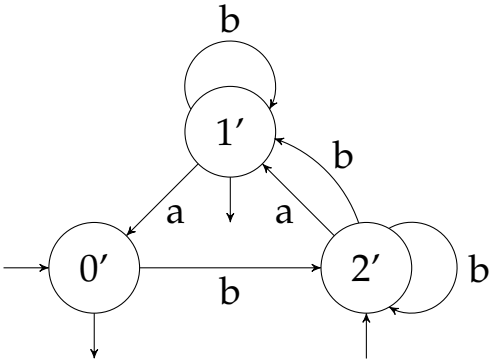
On dira que l'automate $\mathcal{A}$ **accepte** le mot $\mathbf{u}$ s'il contient un tel chemin acceptant pour $\mathbf{u}$.

Concrètement, regardons ce que cela donne si on s'intéresse au graphe $\mathcal{G}$ ci-dessus et aux mots $\mathbf{u} = bbaa$ et $\mathbf{v} = abab$. Dans le premier cas, $\mathcal{G}$ accepte bien le mot $\mathbf{u}$; en effet, la suite de sommets $0, 2, 1, 0, 1$ est bien un chemin acceptant pour $\mathbf{u}$. Au contraire, dans le second cas, $\mathcal{G}$ n'accepte pas le mot $\mathbf{v}$; en effet, un chemin acceptant éventuel devrait nécessairement commencer par le sommet 0, puis se poursuivre avec les sommets $1, 1, 0, 2$, mais nulle arête sortante ne part du sommet 2.

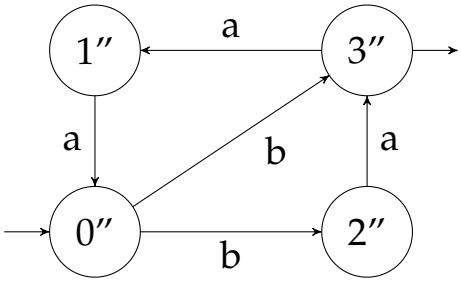
Automates non déterministes, non complets Le lecteur avisé aura remarqué qu'il était très facile de tester si un mot était accepté par l'automate $\mathcal{G}$ présenté ci-

dessus : en effet, il n'existe qu'une seule arête entrante et, pour chaque sommet s et chaque lettre λ de l'alphabet, il existe exactement une arête qui part de s et qui est étiquetée par la lettre λ. On n'a donc jamais de choix à faire, le début du seul chemin acceptant éventuel nous est imposé, et il nous suffit juste de nous promener dans le graphe en suivant un tel début de chemin avant de constater si, oui ou non, on aboutit sur un sommet d'où part une arête sortante.

Cependant, il existe d'autres automates finis moins gentils, tels que les graphes $\mathcal{G}'$ et $\mathcal{G}''$ ci-dessous :



Automate $\mathcal{G}'$



Automate $\mathcal{G}''$

En effet, observons attentivement le graphe $\mathcal{G}'$:

1. depuis le sommet 0, si on veut suivre une arête étiquetée par la lettre a, on est bloqué ;
2. $\mathcal{G}'$ contient deux arêtes entrantes, vers les sommets 0 et 1 ;
3. depuis le sommet 2, en suivant une arête étiquetée par la lettre b, on peut soit aller vers le sommet 1, soit boucler et revenir au sommet 2.

Le point 1 n'est pas trop gênant si le but est simplement de tester si $\mathcal{G}'$ accepte un mot : dans le pire des cas, si on teste un chemin acceptant potentiel et qu'on souhaite prendre une arête qui n'existe pas, c'est juste que notre chemin n'était en fait pas acceptant. Un tel automate est dit **non complet** : c'est un automate où il existe un sommet s et une lettre $\lambda \in \mathcal{A}$ tels que nulle arête étiquetée par λ ne part du sommet s.

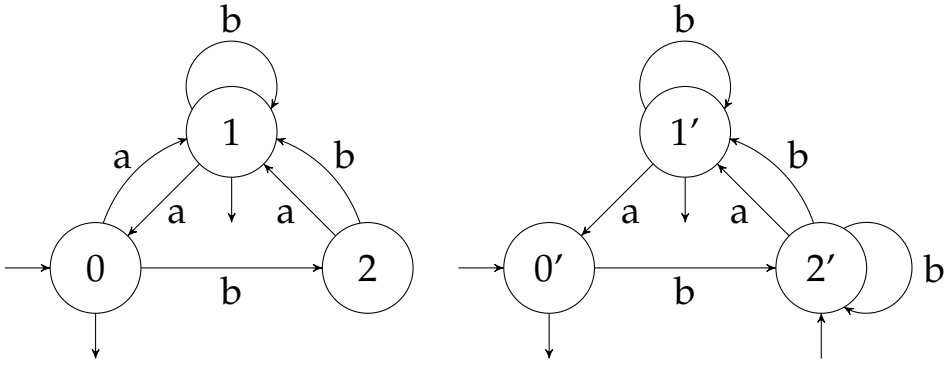
Au contraire, les points 2 et 3 impliquent que l'utilisateur doit tester plusieurs chemins acceptants potentiels : dès l'entrée dans le graphe (point 2) ou à certains moments ultérieurs du parcours (point 3), s'il veut étendre un début de chemin acceptant potentiel, il est obligé de faire des choix a priori entre plusieurs arêtes, sans savoir si un tel choix serait avisé. Notre automate est alors dit **non déterministe** : c'est un automate qui contient

– plusieurs arêtes entrantes, ou bien
– un sommet s et une lettre $\lambda \in \mathcal{A}$ tels qu'il existe plusieurs arêtes étiquetées par λ qui partent du sommet s (et qui vont vers des sommets différents).

De tels automates sont évidemment moins faciles à utiliser, puisque l'on devra parfois s'y reprendre à plusieurs fois pour tester si un mot donné est accepté : peut-être se rendra-t-on compte que le mot est en effet accepté, mais après avoir dû tester un grand nombre de chemins acceptants potentiels ; peut-être devra-t-on, au contraire, tester beaucoup de tels chemins avant de se rendre compte que le mot n'est en fait pas accepté.

Jouer sur les automates et les ensembles associés Pourquoi avoir été chercher des automates a priori aussi peu utilisables que les automates non complets et non déterministes ? Tout simplement parce qu'ils nous permettent de réaliser facilement des opérations assez naturelles sur les ensembles qu'ils représentent.

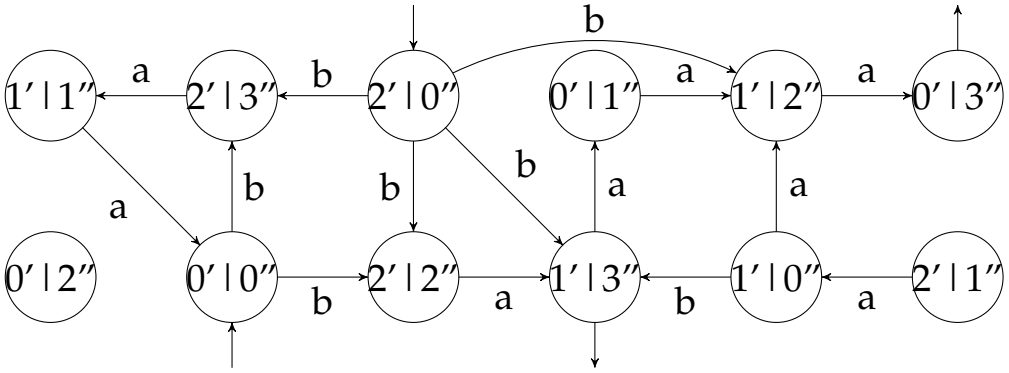
Par exemple, supposons que l'on dispose de deux automates $\mathcal{G}$ et $\mathcal{G}'$, associés à deux ensembles de mots $E_{\mathcal{G}}$ et $E_{\mathcal{G}'}$. On souhaite savoir s'il existe un automate associé à la réunion $E_{\mathcal{G}} \cup E_{\mathcal{G}'}$; si oui, on veut également construire un tel automate. Or, rappelons-nous qu'un mot **u** est accepté par un automate si et seulement si cet automate contient un chemin acceptant pour **u**. Il nous suffirait donc de construire un automate $\mathcal{G} \cup \mathcal{G}'$ dont les chemins acceptants peuvent être identifiés soit à un chemin acceptant de $\mathcal{G}$, soit à un chemin acceptant de $\mathcal{G}'$. Construire un tel automate est donc facile : il suffit de placer côte à côte les deux automates $\mathcal{G}$ et $\mathcal{G}'$ et de prétendre qu'ils forment maintenant, à eux deux, un nouvel automate (non déterministe, puisqu'il contient au moins deux arêtes entrantes).



Automate $\mathcal{G} \cup \mathcal{G}'$

De même, on peut construire un automate $\mathcal{G}' \cap \mathcal{G}''$ dont l'ensemble associé

sera l'intersection $E_{\mathcal{G}'} \cap E_{\mathcal{G}''}$. Pour ce faire, il faut que tout chemin acceptant dans $\mathcal{G}' \cap \mathcal{G}''$ corresponde **simultanément** à **deux** chemins acceptants : l'un dans $\mathcal{G}'$, l'autre dans $\mathcal{G}''$. Il nous faut alors recourir à une ruse, en introduisant un automate **produit** : on imagine que l'on parcourt ces deux chemins en parallèle, et on enregistre, à chaque étape, les états s et s' dans lesquels on se trouve dans notre visite des automates $\mathcal{G}'$ et $\mathcal{G}''$. Tout sommet de l'automate $\mathcal{G}' \cap \mathcal{G}''$ correspond donc à une paire de sommets, et on autorise une arête du type $(s_0|s_0') \to^{\lambda} (s_1|s_1')$ uniquement si les deux arêtes $s_0 \to^{\lambda} s_1$ et $s_0' \to^{\lambda} s_1'$ existent.

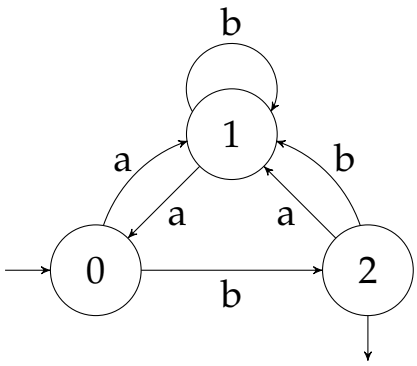


Automate $\mathcal{G}' \cap \mathcal{G}''$

Continuant sur cette voie, on peut réaliser d'autres opérations sur les automates et les ensembles associés. Par exemple, si $\mathcal{G}$ et $\mathcal{G}'$ sont deux automates, on peut construire un automate $\mathcal{G} \cdot \mathcal{G}'$ dont l'ensemble associé est la **concaténation** des deux ensembles $E_{\mathcal{G}}$ et $E_{\mathcal{G}'}$: plus précisément, il s'agit de l'ensemble $E_{\mathcal{G}} \cdot E_{\mathcal{G}'} = \{u_1u_2 \dots u_k v_1 v_2 \dots v_n : u_1u_2 \dots u_k \in E_{\mathcal{G}}, v_1v_2 \dots v_n \in E_{\mathcal{G}'}\}$.
Exercice 1 Construire l'automate $\mathcal{G} \cdot \mathcal{G}'$, où $\mathcal{G}$ et $\mathcal{G}'$ sont les deux automates mentionnés en début de cours.

Cependant, il est très difficile de réussir à créer, dans le cas général, un automate $\neg\mathcal{G}$ qui va représenter le complémentaire de l'ensemble $E_{\mathcal{G}}$ dans l'ensemble $\mathcal{A}^*$ (c'est-à-dire qui accepte un mot **u** si et seulement si $\mathcal{G}$ n'accepte pas **u**). Un espoir est néanmoins que créer un tel automate $\neg\mathcal{G}$ est très facile si $\mathcal{G}$ est déterministe complet. En effet, on a dit, au tout début du cours, qu'on pouvait tester si $\mathcal{G}$ acceptait un mot **u** rien qu'en suivant le seul début de chemin acceptant potentiel pour **u**, puis en vérifiant si, oui ou non, il existe une arête sortante qui part du dernier état dans lequel on est arrivé. Dans ces conditions, pour créer l'automate $\neg\mathcal{G}$, il suffit de considérer l'ensemble S des sommets depuis lesquels part une arête sortante de $\mathcal{G}$, puis de supprimer toute

arête sortante partant d'un sommet de S et de rajouter une arête sortante à tout sommet n'appartenant pas à S.

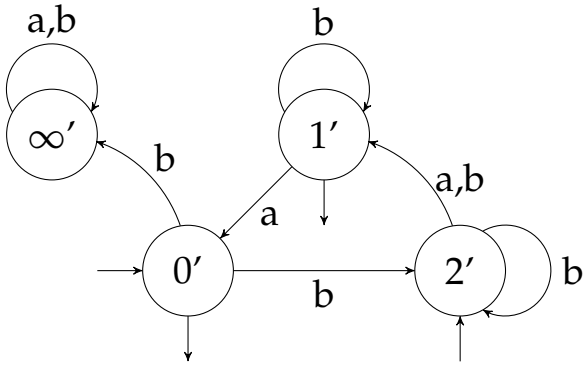


Automate $\neg\mathcal{G}$

Dans ces conditions, être capable de transformer tout automate en un automate déterministe complet équivalent (c'est-à-dire qui est associé au même ensemble de mots) est doublement important :

- cela nous permet d'obtenir des automates avec lesquels il est facile de travailler, par exemple pour tester si un mot est accepté ;
- cela nous permet également de construire un automate associé au complémentaire d'un ensemble $E_{\mathcal{G}}$ quelconque.

Obtenir un automate déterministe complet Montrons d'abord comment, à partir d'un automate $\mathcal{G}$ non complet, on peut obtenir un automate complet $\overline{\mathcal{G}}$ tel que $E_{\mathcal{G}} = E_{\overline{\mathcal{G}}}$. Pour ce faire, c'est assez simple : si on se trouve dans un état s d'où ne part aucune arête étiquetée par la lettre $\lambda \in \mathcal{A}$, il suffit de rajouter une telle arête qui partira vers un état s_∞ par lequel ne peut passer aucun chemin acceptant. Et créer un tel "état poubelle" s_∞ est aisé : pour toute lettre $\lambda \in \mathcal{A}$, on se contente de créer une arête étiquetée par λ, qui part de s_∞ et qui reboucle sur l'état s_∞ lui-même. En outre, notons que, si $\mathcal{G}$ était déjà déterministe, alors $\overline{\mathcal{G}}$ le sera tout autant : une telle remarque ne saurait trop s'apprécier quand on voudra rendre un automate déterministe, puis le rendre complet.



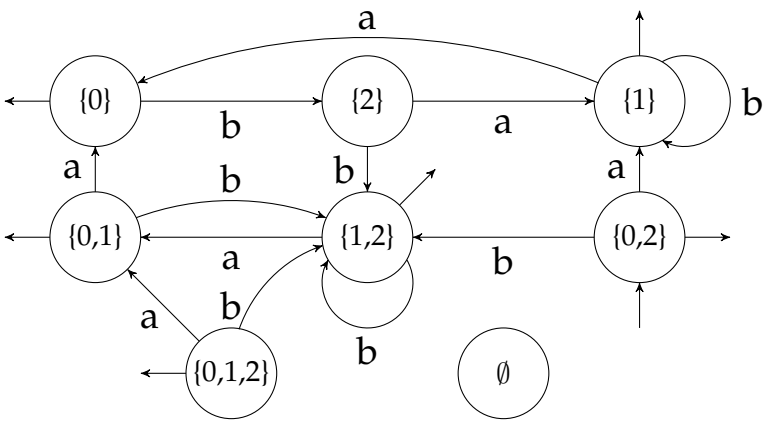
Automate $\overline{\mathcal{G}'}$

Notons d'ailleurs ici que, pour des raisons de lisibilité, on a décidé de fusionner plusieurs arêtes du graphe $\overline{\mathcal{G}'}$ quand on l'a dessiné : il s'agit des arêtes allant du sommet $1'$ au sommet $2'$, et de celles bouclant sur le sommet ∞'. C'est une pratique courante, dont il ne faut donc pas s'émouvoir outre mesure.

Enfin, reste à savoir comment on peut **déterminiser** un automate $\mathcal{G}$ non déterministe. Une idée est que, quand on veut tester si $\mathcal{G}$ accepte un mot **u**, il nous faudrait tester tous les débuts de chemins acceptants potentiels. Cependant, au lieu de faire tous ces tests séquentiellement, l'un après l'autre, rien n'empêche de les faire *en parallèle*. En particulier, si un début de chemin nous amène dans un état s de l'automate, et qu'il nous reste le suffixe $u_k u_{k+1} \dots$ à lire, le fait qu'on ait une chance d'aboutir à un chemin acceptant ne dépend pas du tout des lettres qu'on a déjà lues : seul le fait qu'on ait lu $k-1$ lettres et qu'on soit arrivé dans l'état s est important.

L'idée de la déterminisation est donc de retenir, quand on a lu ces $k-1$ lettres, **l'ensemble des états** de $\mathcal{G}$ dans lesquels on est susceptible de se trouver. Si on aboutit à un ensemble dont un sommet contient une arête sortante, alors $\mathcal{G}$ accepte notre mot **u** ; si on est bloqué par l'absence d'une arête ou si nulle arête sortante ne part d'un sommet de notre ensemble d'arrivée, alors $\mathcal{G}$ n'accepte pas **u**. Cette simple remarque nous permet de construire l'automate des **parties** de $\mathcal{G}$: cet automate déterministe , noté $\det(\mathcal{G})$, est lui aussi associé à l'ensemble $E_{\mathcal{G}}$.

En outre, notons que, si $\mathcal{G}$ est complet, alors $\det(\mathcal{G})$ est lui-aussi complet. Cependant, si on part d'un automate quelconque $\mathcal{G}$, l'automate $\overline{\det(\mathcal{G})}$ obtenu en déterminisant puis en complétant $\mathcal{G}$ contient généralement moins de sommets et d'arêtes que l'automate $\det(\overline{\mathcal{G}})$ obtenu en complétant puis en déterminisant $\mathcal{G}$.



Automate $\det(\mathcal{G}')$

Conclusion On pourrait évidemment se poser bien d'autres questions sur les automates finis. En voici une, non exhaustive :

- tout ensemble de mots est-il associé à un automate ?
- peut-on tester si deux automates sont associés au même ensemble de mots ?
- étant donné un automate $\mathcal{G}$, existe-t-il un unique automate déterministe complet $\mathcal{G}'$ tel que $E_{\mathcal{G}} = E_{\mathcal{G}'}$ et tel que $\mathcal{G}'$ ait un nombre minimal de sommets ?
- et si l'on enlève la contrainte "déterministe complet" ?
- la procédure indiquée ici pour obtenir le complémentaire d'un ensemble est assez compliquée : y a-t-il plus simple ?
- existe-t-il d'autres opérations naturelles sur les ensembles de mots, que l'on pourrait réaliser plus ou moins facilement avec des automates ?

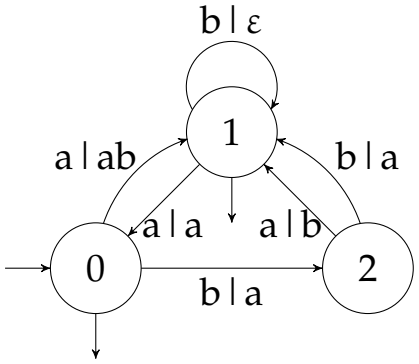
Certaines de ces questions se traitent relativement facilement, si tant est qu'on ait les bonnes idées (ce qui est bien sur la partie difficile du travail). D'autres, au contraire, nécessitent d'utiliser des outils assez lourds. Si vous voulez vous y essayer, et quitte à chercher dans la littérature pour vous aider dans votre quête, n'hésitez pas !

- Transducteurs -

Qu'est-ce qu'un transducteur ? De même qu'il existe de multiples sortes d'automates, il existe bien des sortes de transducteurs. Une fois encore, nous travaillerons sur une petite partie d'entre eux : les transducteurs traitant de **mots finis**.

Comment un transducteur fonctionne-t-il ? Cette fois, on se fixe deux alphabets : un alphabet $\mathcal{A}$ **d'entrée**, et un alphabet $\mathcal{B}$ **de sortie**. Un transducteur ressemble à un automate, sauf que ses étiquettes ne sont plus de simples lettres $\lambda \in \mathcal{A}$. Il s'agit de paires $\lambda|\mathbf{b}$, où λ est une lettre appartenant à $\mathcal{A}$ et $\mathbf{b}$ est un mot fini constitué de lettres de $\mathcal{B}$.

Voici un exemple de transducteur $\mathcal{T}$ pour l'alphabet d'entrée $\mathcal{A} = \{a, b\}$ et l'alphabet de sotie $\mathcal{B} = \{a, b\}$. Notons que $\mathcal{A} = \mathcal{B}$ dans ce cas précis, mais que $\mathcal{A}$ et $\mathcal{B}$ n'ont a priori rien à voir l'un avec l'autre :



Transducteur $\mathcal{T}$

Pourquoi un tel transducteur représente-t-il une fonction ? Considérons un mot fini $\mathbf{u} = u_1 u_2 \dots u_n$, de longueur n, Un **chemin acceptant** pour le mot $\mathbf{u}$ sera une suite de sommets $s_0, s_1, \dots, s_n$ (de longueur $n+1$) telle que

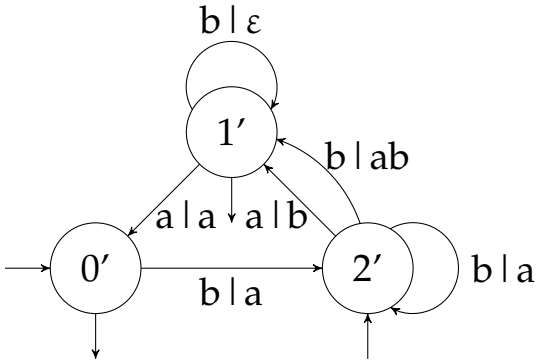
- il existe une arête entrante qui arrive au sommet s_0 ;
- pour tout entier $k \in \{1, 2, \dots, n\}$, il existe une arête normale, étiquetée par une paire $u_k|b_k$ (b_k est un mot quelconque dont les lettres appartiennent à $mathcalB$), qui part du sommet s_{k-1} et arrive au sommet s_k ;
- il existe une arête sortante qui part du sommet s_n.

Le mot $\mathbf{u}$ sera alors associé à la concaténation de mots $b_1 \cdot b_2 \cdot \dots \cdot b_n$.

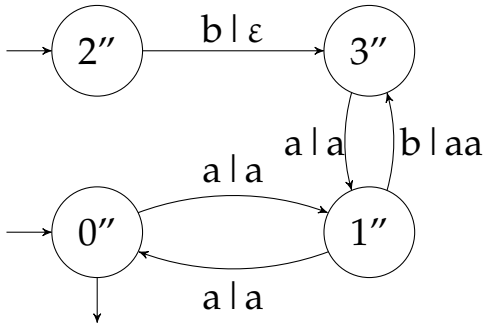
Concrètement, regardons ce que cela donne si on s'intéresse au transducteur $\mathcal{T}$ ci-dessus et aux mots $\mathbf{u} = bbaab$ et $\mathbf{v} = abab$. $\mathbf{u}$ a un unique chemin acceptant dans le transducteur $\mathcal{T}$: ce chemin est la suite de sommets $0, 2, 1, 0, 1, 1$, obtenue en suivant des arêtes d'étiquettes $b|a, b|a, a|a, a|ab, b|\varepsilon$. On associe dont à $\mathbf{u}$ le mot $aaaab$ (le symbole ε est couramment utilisé pour désigner le mot vide, c'est-à-dire le mot qui ne compte aucune lettre). Par contre, $\mathbf{v}$ n'a aucun chemin acceptant, ce qui signifie qu'il n'appartient pas au domaine de définition de la fonction associée à $\mathcal{T}$: on voit déjà là une difficulté, qui est que cette fonction n'est pas nécessairement définie sur $\mathcal{A}^*$ tout entier !

Transducteurs non déterministes, non complets De même qu'il existe des automates non déterministes et non complets, on peut concevoir des transducteurs non déterministes et non complets. La partie "complète" du transducteur se traite aisément : il nous suffit de recréer un état poubelle s_∞, auquel mènent les transitions manquantes du type $\lambda|\varepsilon$.

Cependant, le non déterminisme pose davantage de problèmes : en effet, tout d'abord, un mot **u** doit être associé zéro (s'il n'est pas dans le domaine de définition de la fonction) ou un mot de $\mathcal{B}^*$, mais des chemins acceptants différents peuvent, a priori, donner des mots de $\mathcal{B}^*$ différents. C'est le problème que posent le transducteur $\mathcal{T}'$ suivant, alors même que le transducteur $\mathcal{T}''$ n'est pas touché :



Transducteur $\mathcal{T}'$



Transducteur $\mathcal{T}''$

Exercice 2 Trouver un mot $\mathbf{v} \in \mathcal{A}^*$ auquel le transducteur $\mathcal{T}'$ pourrait associer plusieurs mots **différents**.

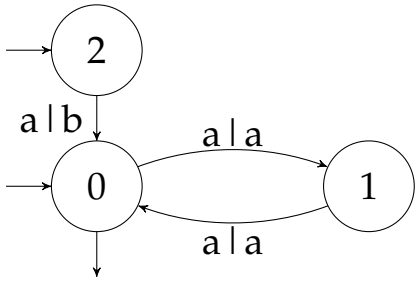
On pourrait naturellement concevoir que le transducteur $\mathcal{T}'$ représente en fait une fonction qui, à tout mot $\mathbf{u} \in \mathcal{A}^*$, associe un **ensemble** de mots $\mathbf{b} \in \mathcal{B}^*$, cet ensemble étant éventuellement vide. Toutefois, puisque nous nous intéressons simplement aux fonctions $f : \mathcal{A}^* \to \mathcal{B}^*$, nous n'étudierons pas ce cas ici ; nous préférerons nous concentrer sur les transducteurs **fonctionnels**, c'est-à-dire les transducteurs qui représentent effectivement une fonction $f : \mathcal{A}^* \to \mathcal{B}^*$ (dont le domaine de définition n'est pas nécessairement $\mathcal{A}^*$ tout entier).

Une fois cette décision prise, une question se pose :

- peut-on tester a priori si un transducteur non déterministe $\mathcal{T}$ est fonctionnel ?
- si oui, peut-on construire un transducteur déterministe $\det(\mathcal{T})$ qui représente la même fonction ?

Il se trouve que la réponse à la première question est **oui**, mais est assez compliquée à expliquer (bien plus que la déterminisation !). Cependant, la réponse

à la deuxième question est **non**, comme l'illustre le contre-exemple suivant, pour l'alphabet de départ $\mathcal{A} = \{a\}$ et l'alphabet d'arrivée $\mathcal{B} = \{a, b\}$:



Transducteur $\mathcal{U}$

En effet, la fonction $f_{\mathcal{U}}$ associée au transducteur $\mathcal{U}$ est telle que $f_{\mathcal{U}}(a^{2k}) = a^{2k}$ et que $f_{\mathcal{U}}(a^{2k+1}) = ba^{2k}$. Ainsi, pour tout mot $\mathbf{u} \in \mathcal{A}^*$, avant d'écrire ne serait-ce que la première lettre de son image $f_{\mathcal{U}}(\mathbf{u})$, il faut déjà connaître la longueur $|\mathbf{u}|$ (mod. 2). Cela fait, il nous faudra alors écrire un mot de longueur $|\mathbf{u}|$, alors même que l'on aura déjà lu toutes les lettres de $\mathbf{u}$. Il nous faut donc savoir combien de lettres écrire, c'est-à-dire retenir la longueur $|\mathbf{u}|$ elle-même. Mais toute notre mémoire à ce moment-là dépendra uniquement de l'état dans lequel on sera. Un transducteur ayant un nombre fini d'états, on ne peut donc distinguer qu'un nombre fini de longueurs de mots, et non pas traiter correctement des mots qui peuvent avoir un nombre arbitrairement grand de longueurs différentes.

Composition des transducteurs Si l'on veut travailler de manière sereine, il faut donc se restreindre à travailler uniquement sur des transducteurs déterministes, ou bien être prêt à renoncer à jamais à tous les avantages que pourrait nous offrir le déterminisme ! Cela dit, quoi qu'il en soit, une des propriétés principales des fonctions est que l'on peut les **composer**.

On cherche donc, étant donnés

– un transducteur $\mathcal{T}$ qui représente une fonction $f_{\mathcal{T}} : \mathcal{B}^* \to \mathcal{C}^*$ et

– un transducteur $\mathcal{T}'$ qui représente une fonction $f_{\mathcal{T}'} : \mathcal{A}^* \to \mathcal{B}^*$,

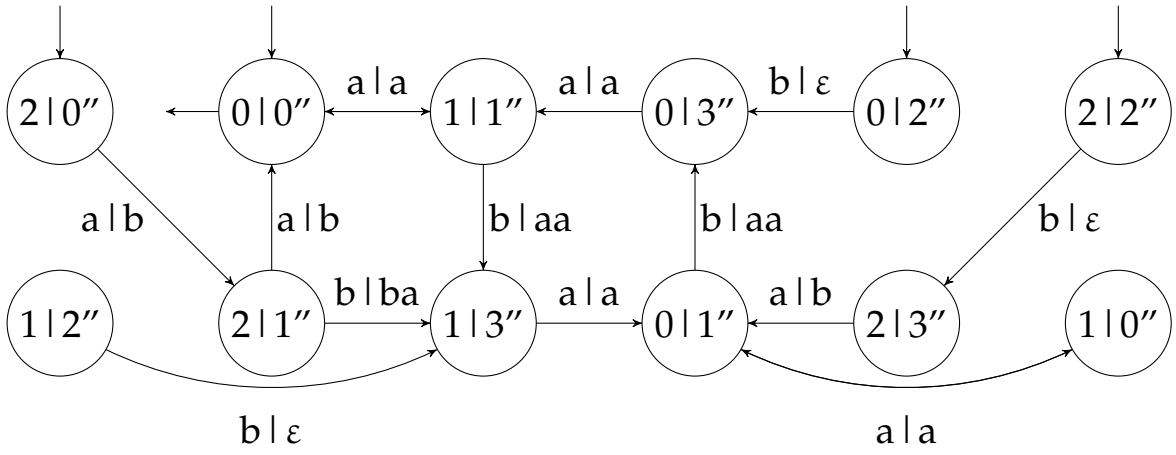
à construire un transducteur $\mathcal{T} \circ \mathcal{T}'$ qui représente la fonction composée $f_{\mathcal{T}} \circ f_{\mathcal{T}'}$.

Pour ce faire, on procède comme suit : supposons que l'on dispose d'un mot $\mathbf{u} = u_1 u_2 \dots u_n \in \mathcal{A}^*$ et que l'on veut calculer $f_{\mathcal{T}} \circ f_{\mathcal{T}'}(\mathbf{u})$. Puisque $\mathcal{T}'$ est fonctionnel, on peut directement se concentrer sur le cas où $\mathcal{T}'$ contient un chemin acceptant $s_0, s_1, \dots, s_n$ pour $\mathbf{u}$, où l'on transite par des arêtes d'étiquettes $u_1|b_1, u_2|b_2, \dots, u_n|b_n$. En outre, $\mathcal{T}$ est fonctionnel lui aussi, donc on se concentre sur le cas où $\mathcal{T}$ contient un chemin acceptant $s'_0, s_1, \dots, s'k$ pour

$f_{\mathcal{T}'}(\mathbf{u}) = \beta_1 \dots \beta_k$, où l'on transite par des arêtes d'étiquettes $\beta_1|c_1, \beta_2|c_2, \dots, \beta_n|c_n$. Notons qu'il faut faire attention, car les b_i sont des mots de longueur a priori quelconque, tandis que les β_i ne sont que des lettres : en particulier, dans le cas général, $\beta_i \neq b_i$.

Au lieu de parcourir les deux transducteurs $\mathcal{T}'$ puis $\mathcal{T}$ l'un après l'autre, on peut alors parcourir les deux transducteurs simultanément, en parallèle : c'est une ruse analogue à celle utilisée pour faire l'intersection de deux automates. Cependant, alors que l'on parcourt une arête du transducteur $\mathcal{T}'$, on peut être amené à créer un nombre quelconque de lettres du "mot intermédiaire " $f_{\mathcal{T}'}(\mathbf{u}) = b_1 \cdot b_2 \cdot \dots \cdot b_n$. Il nous faut donc parcourir d'un seul coup plusieurs arêtes (ou aucune) du transducteur $\mathcal{T}'$; une fois cette subtilité prise en compte, plus aucun problème ne se pose.

C'est ainsi que l'on peut obtenir le transducteur $\mathcal{T}'' \circ \mathcal{U}$ suivant :



Transducteur $\mathcal{T}'' \circ \mathcal{U}$

Conclusion Tout ce qui n'est bien sur qu'une introduction au monde très vaste des transducteurs, qu'il convient d'ailleurs d'étudier conjointement avec leurs cousins les automates. Une fois encore, on pourrait se poser mille et une questions qui n'ont pas été abordées ici, et donc voici quelques unes parmi tant d'autres tout aussi intéressantes :

- étant donnés deux transducteurs fonctionnels $\mathcal{T}$ et $\mathcal{T}'$, sont-ils associés à la même fonction ?
- peut-on tester si $f_{\mathcal{T}}(\mathbf{u}) = f_{\mathcal{T}'}(\mathbf{u})$ pour tout mot $\mathbf{u}$?
- si $\mathcal{T}$ est déterministe, existe-t-il un unique transducteur déterministe complet $\mathcal{U}$ tel que $f_{\mathcal{T}} = f_{\mathcal{U}}$ et tel que $\mathcal{U}$ ait un nombre minimal de sommets ?
- et si l'on enlève la contrainte "déterministe complet" ?

– existe-t-il d'autres opérations naturelles sur les fonctions, que l'on pourrait réaliser plus ou moins facilement avec des transducteurs ?

Une fois encore, certaines de ces questions trouvent facilement une réponse, tandis que d'autres sont probablement ouvertes à l'heure actuelle (selon le moment auquel vous lirez ce texte) : dans tous les cas, faciles ou difficiles, elles n'attendent que vous pour être résolues !