# Extraction d'information avec un transducteur

F. Barthélemy

1er janvier 2017

Cet exemple illustre comment un transducteur peut extraire toutes les occurrences d'un certain motif trouvées dans une chaîne ou un ensemble de chaînes. Cet exemple est très simple pour que les transducteurs puissent être dessinés et compris sous forme graphique. On réduit l'alphabet pour diminuer le nombre de flèches.

Le principe de l'extraction d'information : on recherche un certain motif (pattern) dans la chaîne. Ce motif est réécrit à l'identique alors que tout le reste de la chaîne est éliminé en étant réécrit en epsilon.

Dans notre exemple, il s'agit d'extraire tous les chiffres présents dans la chaîne. Pour ce faire, on réécrit un chiffre et on élimine tout ce qu'il y a avant et tout ce qu'il y a après.

En opengrm, cela s'écrit comme suit :

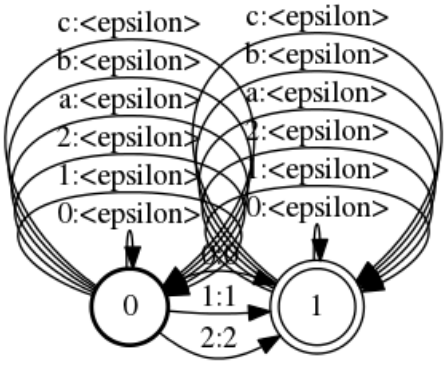
```
export alphabet = Optimize["0"|"1"|"2"|"a"|"b"|"c"];

export chiffre = Optimize["0"|"1"|"2"];

export chiffre_id = Optimize[("0":"0")|("1":"1")|("2":"2")];

export chiffre_picker = Optimize[(alphabet:"")* chiffre_id ((alphabet:"")*)];
```

Le transducteur `chiffre_picker` est le suivant :




L'opération pour appliquer ce transducteur à une chaîne est la *sélection* qui en opengrm, s'écrit comme la composition avec @.

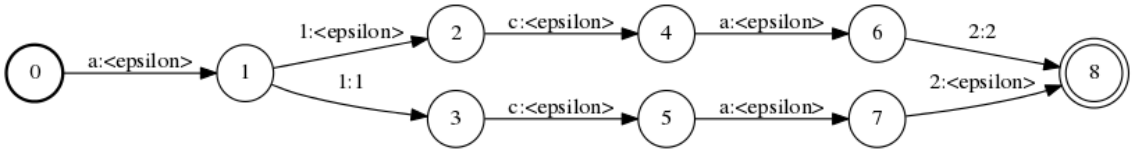
```
export exemple = "a1ca2";

export selection = Optimize[exemple @ chiffre_picker];

export chiffres = Optimize[Project[selection,'output']];
```

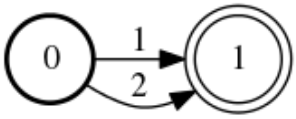
La sélection va produire un transducteur avec un chemin succès différent pour chaque occurrence d'un chiffre dans la chaîne. Dans la chaîne `exemple`, il y a deux chiffres : il y a donc deux chemins succès dans `selection`.

Voici le transducteur `selection` :




Ensuite, lorsqu'on projette sur la sortie et qu'on optimize, les epsilon sont éliminés et il ne reste que l'ensemble des chiffres.

Voici l'automate `chiffres` :




Notez bien que les différents chemins correspondent à différentes chaînes du langage. Le résultat de l'extraction est donc l'ensemble des occurrences du motif. Il y a implicitement un *ou* entre ces différentes occurrences.

Par exemple, l'automate `chiffres` pourrait s'écrire en opengrm :

```
export chiffres = Optimize["1"|"2"];
```