TD 2 - Les automates

Retrouvez tous les énoncés et les corrections des TP sur ma page personnelle :

`http://perso.ens-lyon.fr/hadrien.croubois/`

Important

- Jusqu'à preuve du contraire une fonction ne marche pas. Testez vos fonctions !
- N'hésitez pas à utiliser les fonctions de la librairie standard, notamment `mem`, `map`, `flat_map`, `intersect`, `exists`, `subtract` ...[1]

Utile pour la suite

- Écrivez une fonction `split_int:int -> int list` qui à tout entier n associe les chiffres de n en base 10 de gauche à droite.

De l'implémentation des automates non-déterministes

On se propose de définir les automates non déterministe sans ϵ-transition de la façon suivante :

```
type ('a, 'b) automaton =
{
    starts : 'a list;
    accepts : 'a list;
    trans : 'a -> 'b -> 'a list;
};;
```

- Quels sont selon vous les avantages et les inconvénients d'une telle implémentation ?
- Écrire une fonction `gen_auto_length_div:int -> (int, 'a) automaton` décrivant un automate reconnaissant les mots dont la longueur est un multiple de n.
- Écrire une fonction `gen_auto_length_div:int -> (int, 'a) automaton` décrivant un automate reconnaissant les mots sur l'alphabet $\{0-9\}$ qui sont des multiples de n.
- Implémentez une fonction `run:('a, 'b) automaton -> 'b list -> bool` qui détermine si un mot est accepté par l'automate.
- Discuter les performances de votre fonction. Pourriez vous les améliorer ?

Vers une autre implémentation

On se propose de redéfinir les automates sans ϵ-transition à l'aide d'une liste d'adjacence :

```
type ('a, 'b) automaton =
{
    starts : 'a list;
    accepts : 'a list;
    trans : (('a * 'b) * 'a) list;
};;
```

[1] Operations on lists : http://caml.inria.fr/pub/docs/manual-caml-light/node14.13.html

- Quels sont selon vous les avantages et les inconvénients d'une telle implémentation ?
- Réécrivez les fonctions `gen_auto_length_div` et `gen_auto_div` pour le nouveau format d'automate.
- Implémentez une fonction `run:('a, 'b) automaton -> 'b list -> bool` qui sous l'hypothèse que `starts` est de taille 1 et que l'automate est déterministe, détermine si un mot est accepté par l'automate.
- Écrire une fonction `multi_trans:('a, 'b) automaton -> 'b -> 'a list -> 'a list` qui à un automate $\mathcal{A}$ dont l'ensemble de transitions est τ, à un caractère c de l'alphabet et à un ensemble d'état E associe

  $$\tilde{\tau}(E) = \{y | \exists((x, c), y) \in \tau, x \in E\}$$

  Autrement dit, `multi_trans auto c set` est l'ensemble des états atteint en lisant `c` depuis un états de `set`.
- Écrire une fonction `auto_alphabet:('a, 'b) automaton -> 'b list` qui renvoi l'ensemble des caractères qui apparaissent dans la liste des transitions de $\mathcal{A}$.
- Rappeler l'algorithme des partie permettant déterminiser un automate.
- Implémenter cet algorithme sous la forme d'une fonction :

  `build_deterministic:('a, 'b) automaton -> ('a list, 'b) automaton`