FSA - Mode d'emploi

Le logiciel FSA permet de manipuler des automates finis. Plus précisément, il permet de :
- définir des reconnaisseurs et des transducteurs à partir d'expressions régulières
- transformer des reconnaisseurs et des transducteurs (déterminisation, minimisation, élimination des transitions vides ...)
- tester la reconnaissance d'un mot par un reconnaisseur et appliquer un transducteur à un mot pour produire des mots en sortie
- visualiser des automates

Le logiciel peut être utilisé selon deux modes :
- En mode d'interprétation de commande, il faut alors de lancer avec la commande `/usr/public/fsa/fsa`.
- En mode graphique, il faut alors de lancer avec la commande `/usr/public/fsa/fsa -tk`.

1 Définition d'automates à partir d'expressions régulières

FSA permet de décrire des reconnaisseurs et des transducteurs à partir d'expressions régulières. Ces dernières sont composées d'expressions élémentaires et d'opérateurs. Les opérateurs sont divisés en trois groupes, les opérateurs standards permettant de définir des reconnaisseurs, les opérateurs supplémentaires qui permettent d'alléger la définition de certains reconnaisseurs et les opérateurs de définition de transducteurs.

1.1 Expressions élémentaires

Les expressions élémentaires sont les briques de base à partir desquelles sont construites les expressions régulières. Une expression élémentaire est :
- un caractère ASCII. Si ce dernier est une majuscule ou un caractère spécial, il doit être mis entre apostrophes.
  Le symbole ? dénote l'ensemble de tous les caractères ASCII.
- une macro expression

1.2 Opérateurs standards

- Union `{A, B}`
- Concaténation `[A, B]`
- Itération `A *`

- Intersection `A & B`
- Complémentation `~ A`

1.3 Opérateurs supplémentaires

- Différence `A - B`
- `$E` L'ensemble des mots ayant une partie appartenant à E, équivalent à `[?*,E,?*]`.
- `E+` au moins une fois, équivalent à `[E, E*]`.
- `E^` zéro ou une fois, équivalent à {`[]`, `E`}.
- `E..T` où `E` et `T` sont des caractères ASCII. Cette expression dénote l'ensemble caractères dont le code ASCII est compris entre celui de `E` et de `T`)

1.4 Opérateurs de transduction

- Produit cartésien `A x B` (`A` et `B` sont des reconnaisseurs)
- Composition `A o B` (`A` et `B` sont des transducteurs)
- `domain(E)` première projection du transducteur `E` (le reconnaisseur du langage d'entrée de `E`)
- `range(E)` deuxième projection du transducteur `E` (le reconnaisseur du langage de sortie de `E`)

1.5 Transformation de reconnaisseurs en transducteurs

Lorsqu'un reconnaisseur R apparaît là où devrait apparaître un transducteur, il est transformé en un transducteur qui associe chaque mot du langage reconnu par R avec lui-même.

Exemple : dans l'expression {`a,b`} `o (b x c)` les deux opérandes de l'opérateur de composition (`o`) devant être des transducteurs, le reconnaisseur {`a,b`} est transformé en un transducteur ({`a x a, b x b`})

2 Macro expressions

FSA permet de définir des macro expressions dans un fichier séparé que l'on peut charger dans FSA à l'aide de la fonction `LoadAux` du menu `File`. Les fichiers de définition des macros doivent avoir l'extension `.pl`. Le fichier de définition des macros doit commencer par la commande suivante : `:- multifile macro/2.`.

La définition d'une macro constante (macro sans paramètres) se fait de la manière suivante : `macro(x,y)` où `x` est le nom de la macro et `y` une expression régulière qui constitue la valeur de la macro. Exemple : `macro(voyelles,{a,e,i,o,u,y}).` (*Attention de ne pas oublier le point à la fin de la définition de la macro !*)

Une macro peut être paramétrée par des variables, comme dans l'exemple suivant : `macro(etoile(E), (E)*)`. Les identificateurs des variables doivent commencer par une majuscule.

Si l'on veut utiliser dans un fichier `f2.pl` les macros définies dans le fichier `f1.pl`, il faut ajouter dans `f2.pl` la commande `:- ensure_loaded('f1').`