Projet individuel d'algorithmique-programmation IPF : groupe 4

1er mars 2017

1 Informations générales

1.1 Travail à rendre

Le projet est à réaliser en OCaml **individuellement**. Il sera accompagné d'un **dossier** contenant impérativement la description des choix faits, la description des types et des fonctions. Pour chaque fonction, on donnera impérativement l'interface complète (dans le code en commentaire et dans le rapport pour les fonctions présentées).

Même si le sujet est décomposé en questions, il est possible qu'une question se résolve par l'écriture d'une ou plusieurs fonctions intermédiaires. Celles-ci doivent comporter une interface également.

Le dossier fournira également des cas de tests accompagnés des résultats attendus et retournés.

Sur le site du cours figure un petit document sur ce que l'on attend dans un rapport. Consultez-le !

Attention, le code doit être purement fonctionnel : pas de boucle, pas d'affectation, pas de tableau. La seule tolérance concerne les affichages (voir document en ligne).

1.2 Calendrier et procédure de remise

Le projet est à rendre le **mercredi 29 mars 2017** minuit au plus tard sur le site de dépots `exam.ensiie.fr`. Cliquez sur **IPF_S2_2017**. Vous y déposerez une archive contenant votre rapport au format pdf (impérativement) et le code de votre projet.

2 Énoncé : Machines de Turing

Les machines de Turing sont un modèle abstrait de calcul. Le but du projet est d'implémenter les machines de Turing.

2.1 Partie 1 : Machines de Turing à un ruban

Les machines de Turing sont constituées d'un ou plusieurs rubans, d'un ensemble fini d'états, et d'une table de transition. On se concentre pour l'instant sur les machines à un ruban.

Un ruban modélise un mémoire infinie à gauche et à droite, constituée de cases contenant des caractères, et sur laquelle est placée une tête de lecture/écriture. La tête peut se déplacer à gauche ou à droite sur le ruban. On distingue un caractère spécial, le caractère blanc. La table de transition indique, étant donnés un état et le caractère sous la tête de lecture du ruban, quel caractère sera écrit sous la tête de lecture, dans quel état on arrive, et dans quelle direction la tête de lecture se déplace (d'une case).

Plus formellement, une machine de Turing est un 5-uplet $\langle \Sigma, Q, q_i, \delta, q_f \rangle$ où Σ est le vocabulaire, c'est-à-dire l'ensemble des caractères qui contient donc le caractère blanc qu'on notera `B` ; Q est l'ensemble fini des

états ; q_i est l'état initial de la machine ; q_f l'état final, ou état d'arrêt ; $\delta : (Q \times \Sigma) \to (Q \times \Sigma \times \{G, D\})$ est la fonction de transition qui à un couple (état courant, caractère lu) associe un triplet (état suivant, caractère écrit, direction).

Le fonctionnement d'une machine de Turing sur un ruban initial est le suivant : au départ, la machine est dans l'état initial. Puis, tant qu'on atteint pas l'état d'arrêt, on regarde le caractère sous la tête de lecture du ruban et on regarde ce que la fonction de transition dit pour l'état courant et le caractère : l'état devient l'état suivant, on écrit le caractère sur le ruban à la position de la tête de lecture et on déplace ensuite cette tête d'une case dans la direction indiquée.

Par exemple, si l'état courant est q_0 et que le ruban est

⋯	B	1	B	B	1	B	B	1	1	B	1	⋯
						↑						

et que la fonction de transition associe à (q_0, B) le triplet $(q_1, 1, G)$, on arrive dans l'état q_1 avec le ruban

⋯	B	1	B	B	1	1	B	1	1	B	1	⋯
					↑							

On utilise le type suivant pour les rubans :

```
type ruban = Tete of char list * char * char list
```

L'élément du milieu, de type `char` est le caractère situé sous la tête de lecture/écriture ; celui de gauche est la liste des caractères situés à gauche de la tête, en allant de droite à gauche (Autrement dit, le premier élément de cette liste est le voisin de gauche du caractère sous la tête.) ; celui de droite est la liste des caractères situés à droite de la tête, en allant de gauche à droite. On considérera que les caractères au delà de la fin des listes sont tous blancs. On utilisera le caractère `'B'` pour les blancs.

1. Définir le ruban `blancs` qui ne contient que des blancs.
2. Écrire deux fonctions `deplace_gauche` et `deplace_droite` qui prennent en paramètre un ruban et qui retourne le ruban obtenu en déplaçant la tête respectivement vers la gauche et vers la droite. On sera amené à rajouter des caractères blancs dans le cas où on atteint la limite des listes représentant le ruban.
3. Écrire une fonction `affiche_ruban` qui prend un ruban et qui l'affiche sur la sortie standard. On affichera le caractère situé sous la tête entre crochets. Le premier exemple ci-dessus s'affichera donc :

   ```
   B 1 B B 1 [B] B 1 1 B 1
   ```

   On va implémenter la fonction de transition à l'aide de listes d'associations. Une liste d'association est une liste de couples associant à des clefs des valeurs. Quand on recherche une clef dans une liste d'association, on retourne la deuxième composante du premier couple de la liste dont la première composante est la clef recherchée.
4. Écrire une fonction `rechercher` qui prend en paramètre une liste d'association et une clef et qui retourne la première valeur associée à la clef dans la liste. On lèvera l'exception `Not_found` si la clef n'est pas présente dans la liste.
5. Définir un type pour les directions (G ou D).
6. Une machine de Turing sera représenté par un triplet : l'état initial ; la fonction de transition décrite par une liste d'association dont les clefs sont des couples d'état et de caractère et dont les valeurs sont des triplets d'état, de caractère et de direction ; l'état final.
   Définir un type pour les machines de Turing, en le choisissant polymorphe par rapport au type des états.
7. Écrire une fonction `pas` qui prend en paramètre une machine de Turing, un ruban et un état et qui retourne le couple de l'état et du ruban obtenu après avoir fait une étape de transition de la machine de Turing en partant de l'état et du ruban donnés.

8. Écrire une fonction `execute` qui prend en paramètre une machine de Turing et un ruban et qui exécute la machine de Turing sur le ruban jusqu'à ce qu'on atteigne l'état final.
9. Représenter les machines de Turing suivantes :

   Castor affairé :
   — État initial : a
   — État final : h
   — Table de transition :

a	B	b	1	D
a	1	c	1	G
b	B	a	1	G
b	1	b	1	D
c	B	b	1	G
c	1	h	1	D

   Mystère :
   — État initial : 1
   — État final : 0
   — Table de transition :

1	B	0	B	D
1	1	2	B	D
2	B	3	B	D
2	1	2	1	D
3	B	4	1	G
3	1	3	1	D
4	B	5	B	G
4	1	4	1	G
5	B	1	1	D
5	1	5	1	G

10. Tester l'exécution du castor affairé sur le ruban vide.
11. Tester l'exécution de la machine mystère sur un ruban contenant n 1 consécutifs, pour un certain nombre n. Que fait la fonction mystère ?

2.2 Partie 2 (Bonus) : Machine de Turing à deux rubans

Cette partie ne pourra être abordée que si la première partie fonctionne correctement.

On considère maintenant une machine à deux rubans. Dans ce cas, la fonction de transition associe à un état et aux deux caractères sous les têtes de lecture des rubans, un nouvel état, et des caractères et des directions de déplacement pour chacun des deux rubans.

1. Définir un nouveau type pour les machines à deux rubans, et écrire les fonctions `pas2` et `execute2` correspondantes.
2. Écrire une fonction `parallelise` qui prend deux machines de Turing à un ruban et qui retourne la machine de Turing à deux rubans qui effectue les actions des deux machines en parallèle sur les deux rubans. On notera l'intérêt d'avoir choisi un type d'état polymorphe.
3. Tester en parallélisant les deux machines de la première partie.