info401 : Programmation fonctionnelle
TP 0 : prise en main de Caml

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22, poste : 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/

Le but de ce TP est de vous familiariser avec Caml et l'interface que nous allons utiliser pour les TP suivants. Il n'est pas demandé de compte-rendu, mais vous pouvez m'envoyer ce que vous avez fait pour avoir des commentaires.

Partie 0 : Préliminaires

Question 1. Redémarrez votre ordinateur, et choisissez `Linux` au démarrage. Si vous n'avez pas l'habitude de Linux, passez quelques minutes pour explorer l'interface. Vous avez accès à vos dossiers personnels, au web (le navigateur internet s'appelle `iceweasel`) etc.

...

...

Partie 1 : L'interprète Caml simple

Le plus simple pour expérimenter avec des tous petits programmes Caml est d'utiliser l'interprète Caml. Pour lancer cet interprète, vous avez plusieurs solutions.

- La méthode simple : vous allez sur ma page web pour télécharger le petit fichier `interprete-caml` que vous sauvegardez sur votre bureau, puis vous double-cliquez sur l'icône correspondante.
- La méthode pour les experts : vous lancer l'application `Terminal` (`Applications --> Accessoires --> Terminal`), puis vous tapez la commande suivante à l'intérieur de ce terminal :*

```
$ ledit -x -h ~/.ocaml_history ocaml
```

Une fois dans l'interprète Caml, vous pouvez :

- taper des morceau de Caml (n'oubliez pas les "`;;`" pour demander l'évaluation de ce que vous avez écrit),
- oublier ce que vous avez écrit depuis le dernier "`;;`" en appuyant simultanément sur "Control" et sur "c",
- retrouver des lignes plus vieilles avec la flèche du haut,
- aller au début de la ligne en appuyant sur "Control" et "a", ou aller à la fin de la ligne avec "Control" et "e",
- quitter l'interprète avec la commande "`#quit ;;`". (N'oubliez pas le "`#`".)

Question 1. Reprenez les exercices que l'on a fait en TD et testez dans l'interprète Caml. Vous avez le droit (c'est même encouragé) de faire des trucs bizarres et interdits, mais essayez de comprendre ce qui se passe.

Question 2. Dans le code suivant, quelle sera la valeur donnée par `f 2` la première et la deuxième fois. Expliquer ce qu'il se passe...

```
let x=4 ;;
let f y = y+x ;;
f 2 ;;
```

* vous pouvez aussi utilisez la commande `$ ocaml`, mais vous n'aurez pas la possibilité de vous déplacer avec les flèches...

```
let x=5 ;;
f 2 ;;
```

Question 3. Que se passe-t'il si vous ne mettez pas de `else` dans un `if` ?
Comment expliquez vous cela ?

Partie 2 : Emacs et Caml

L'interprète tout seul est un peu pauvre, et si l'on veut faire des programme un peu gros, il faut sauvegarder ce que l'on fait dans un fichier. Vous avez peut-être déjà utilisé l'éditeur de texte "GEdit". Pour ce cours, nous utiliser l'éditeur de texte "Emacs", qui a plusieurs avantages.

Comme l'utilisation de Emacs est un peu étrange au début, et pour éviter de faire un sujet de TP de 15 pages, je vous montrerais comment faire les choses suivantes pour commencer à programmer en Caml.

- Étape préliminaire : créez un répertoire "`TP-info-401`",
- lancez "Emacs" (`Applications --> Accessoires --> Emacs`) et créez un nouveau fichier (`File --> Visit New File` ; ou premier icône en haut à gauche) appelé "`tp1.ml`" dans le répertoire crée au dessus. *Attention,* il faut donner le nom de votre fichier sur la ligne en bas de la fenêtre :
  - . commencer par taper le début du nom du répertoire que vous voulez utiliser, puis appuyer sur la touche "Tabulation" : le nom du répertoire se terminera tout seul,
  - . tapez le nom du fichier (`tp1.ml`) et validez.
- Ajoutez au moins les options suivantes :
  - . `Options --> Syntax Highlighting`,
  - . `Options --> Paren Match Highlighting`,
  - . `Options --> Active Region Highlighting`,
  - . et sauvegardez ces options (`Options --> Save Options`).
- Tapez le code Caml suivant dans votre fichier : (remarquez que Emacs colorie les mots clés, et l'utilisation de la touche Tabulation n'a pas l'effet attendu)

```
let max (a:int) (b:int) : int =
  if (a<b)
  then b
  else a
```

- Faites un copier-coller dans l'interprète Caml (pour copier, il suffit de sélectionner, et pour coller, il suffit appuyer sur le bouton du milieu),
- encore mieux, mais vous pouvez également utilisez `Tuareg --> Interactive Mode --> Evaluate Buffer` pour donner tout ce que vous avez écrit à un interprète Caml qui se retrouvera dans une sous-fenêtre de Emacs.
- passez un moment à essayer différentes chose avec Emacs et Caml.

Ça parait un peu complexe au début, mais vous verrez que c'est finalement bien pratique.

Question 1. En utilisant Emacs et le mode interactif, faites les questions 2, 3 et 4 de l'exercice 2 du TD 1 en écrivant les fonctions suivantes :

- `fact : int -> int` (quelle est la plus grande valeur que vous pouvez calculer ?),
- `somme_carre : int -> int`,
- `somme_fonction : ???`.

Question 2. On peut définir les nombres pairs et impairs de la manière suivante :

- 0 est pair,
- $n > 0$ est pair si et seulement si $n - 1$ est impair,
- $n > 0$ est impair si et seulement si $n - 1$ est pair.

En utilisant directement cette définition, écrivez, en Caml, deux fonctions `pair` et `impair` de type `int -> bool` qui renvoient `true` ou `false` suivant que leur argument est pair ou impair. Vous n'avez pas le droit d'utiliser d'opérations arithmétiques autre que la soustraction `n-1` .

Question 3. La suite de Fibonacci est définie de la manière suivante :

$$u_0 = 0 \qquad u_1 = 1 \qquad u_{n+2} = u_{n+1} + u_n$$

Programmez une fonction `fib : int -> int` pour la calculer.

Quelle est la plus grande valeur que vous pouvez obtenir ?

À votre avis, combien d'additions sont calculées lors du du calcul de `fib 10` ? Qu'en pensez-vous ?

Question 4. Écrivez deux fonctions sur les listes :
- `taille` qui calcule la taille d'une liste,
- `alterne` qui prend un élément sur deux une liste.
  (Par exemple `alterne [1 ;2 ;3 ;4 ;5]` donne `[1 ; 3 ; 5]`.)

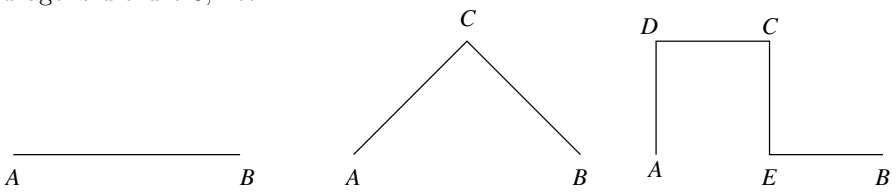
Partie 3 : un petit peu de dessin

La courbe du dragon est obtenue en suivant le processus suivant :
- le dragon d'ordre 0 sur le segment AB est simplement AB
- le dragon d'ordre $n+1$ sur le segment AB s'obtient en :
  . plaçant un coin C à gauche de AB, de manière à ce que l'angle $\widehat{ACB}$ soit un angle droit.
  . en faisant un dragon d'ordre $n-1$ sur les segments AC et BC (attention à l'ordre des points).

Les coordonnées du point C sont

$$x_C = \frac{x_A + x_B}{2} + \frac{y_B - y_A}{2} \quad \text{et} \quad y_C = \frac{y_A + y_B}{2} + \frac{x_A - x_B}{2}$$

Voici les dragons d'ordre 0,1 et 2 :



Question 1. Programmer une fonction `dragon : int -> ...` qui dessine la courbe du dragon d'ordre n (premier argument) pour un segment AB (autre argument de la fonction).

▷ Pour pouvoir dessiner, vous aurez besoins de :
- mettre un "`#load "graphics.cma";;`" au début de votre fichier,
- mettre un "`open Graphics;;`" au début de votre fichier (après la ligne précédente),
- ouvrir une fenêtre graphique avec "`open_graph " 800x600";;` (n'oubliez pas l'espace avant le `800`). La fenêtre aura une taille de 800 par 600 pixels, et le coin en bas à gauche aura comme coordonnées $(0, 0)$.

▷ Quand des expressions Caml produisent des effets de bord, on peut les séquentialiser avec un point virgule :

```
expr1 ;
expr2 ;
expr3
```

▷ Pour tracer un segment de droite entre les points de coordonnées `xa,ya` et `xb,yb`, il faut utiliser

```
moveto xa ya ;
lineto xb yb
```

(Notez l'utilisation du point virgule.)

▷ Pour effacer le contenu de la fenêtre graphique, vous pouvez utiliser

```
clear_graph ()
```

Vous pouvez vous reporter à la documentation en ligne (lien sur le wiki) pour les fonctions supplémentaires. (Vous n'en avez pas besoin pour ce TP.)

Remarque : l'utilisation du point virgule pour séquentialiser des expression n'a d'intérêt que lorsque les expressions sont impures. (Elle modifient quelque chose : l'état de la fenêtre graphique en l'occurrence.)

Question 2. Que pensez vous de la précision des dessins ?