Contents

1 Module `Exemple` : Plot : code ml pour ocamlc

Plot : Projet MOcaml

ENS de Cachan - 2012 L3 Informatique Projet MOcaml v1.0.4

Besson Lilian. InspirÃ© de 'meta_plot.c' et 'script_metaplot.sh' Ã©crits par Lilian Besson en FÃ©vrier 2012. (c) Naereen Corp.

Utilise le module ANSITerminal.

2 Module `MOcamlPlot` : Plot : code ml pour ocamlc

Plot : Projet MOcaml

ENS de Cachan - 2012 L3 Informatique Projet MOcaml v1.0.4

Besson Lilian. InspirÃ© de 'meta_plot.c' et 'script_metaplot.sh' Ã©crits par Lilian Besson en FÃ©vrier 2012. (c) Naereen Corp.

Utilise le module ANSITerminal.

```
module Top :
  sig
    type color = ANSITerminal.color =
      | Black
      | Red
      | Green
      | Yellow
      | Blue
      | Magenta
      | Cyan
      | White
      | Default
    type style = ANSITerminal.style =
      | Reset
      | Bold
      | Underlined
      | Blink
      | Inverse
      | Hidden
```

```
  | Foreground of color
  | Background of color
val black : style
val red : style
val green : style
val yellow : style
val blue : style
val magenta : style
val cyan : style
val white : style
val default : style
val on_black : style
val on_red : style
val on_green : style
val on_yellow : style
val on_blue : style
val on_magenta : style
val on_cyan : style
val on_white : style
val on_default : style
val set_autoreset : bool -> unit
val print_string : style list -> string -> unit
val prerr_string : style list -> string -> unit
val printf :
  style list ->
  ('a, unit, string, unit) Pervasives.format4 -> 'a
type loc = ANSITerminal.loc =
  | Eol
  | Above
  | Below
  | Screen
val erase : loc -> unit
val set_cursor : int -> int -> unit
val move_cursor : int -> int -> unit
val move_bol : unit -> unit
val pos_cursor : unit -> int * int
val save_cursor : unit -> unit
val restore_cursor : unit -> unit
val resize : int -> int -> unit
```

```
    val size : unit -> int * int
    val scroll : int -> unit
  end
val line : unit -> int
val column : unit -> int
```

RÃ©cupÃ¨re la taille de la fenÃ¨tre ANSI. Ne fonctionne pas lors de l'utilisation d'un wrapper comme "ledit" ou "rlwrap". Par dÃ©faut, line = 36; column = 135, rÃ©solution de mon terminal en plein Ã©cran.

L'amÃ©lioration du module ANSITerminal est en cours pour rÃ©gler Ã§a. C'est pas facile !

```
val arrayfloat_of_arrayint : int array -> float array
```

Change un tableau d'entiers en tableau de flottants.

```
val bool_of_int : int -> bool
val int_of_bool : bool -> int
```

Conversions entiers et boolÃ©ens.

```
val bool_of_float : float -> bool
val float_of_bool : bool -> float
```

Conversions entiers et flottants.

```
val maxof_array : 'a array -> 'a
```

Calcul du max.

```
val minof_array : 'a array -> 'a
```

Calcul du min.

```
exception Mauvaise_valeur_gauche
```

Si le programme cherche Ã Ã©valuer une valeur trop a gauche (bug).

```
exception Mauvaise_valeur_droite
```

Si le programme cherche Ã Ã©valuer une valeur trop a droite (bug).

```
exception Dessin_impossible
```

Si les paramÃ¨tres du tracÃ© font qu'il est impossible a rÃ©aliser. Par exemple si `maxx` < `minx` ou `maxy` < `miny`.

```
val getchar : unit -> char
```

Comme le getchar du C.

```
val plot_array :
  vect:float array ->
  ?title:string ->
  ?param:bool ->
  ?mon_out:Pervasives.out_channel ->
  ?minx:float ->
  ?maxx:float -> ?miny:float -> ?maxy:float -> ?full_ascii:bool -> unit -> unit
```

Affiche les valeurs d'un tableau.

```
val plot_list :
  liste:float list ->
  ?title:string ->
  ?param:bool ->
  ?mon_out:Pervasives.out_channel ->
  ?minx:float ->
  ?maxx:float -> ?miny:float -> ?maxy:float -> ?full_ascii:bool -> unit -> unit
```

Idem, mais avec une liste.

```
val plot_array_int :
  vect:int array ->
  ?title:string ->
  ?param:bool ->
  ?mon_out:Pervasives.out_channel ->
  ?minx:float ->
  ?maxx:float -> ?miny:float -> ?maxy:float -> ?full_ascii:bool -> unit -> unit
```

Idem, mais avec un tableau d'entiers.

```
val plot_list_int :
  liste:int list ->
  ?title:string ->
  ?param:bool ->
  ?mon_out:Pervasives.out_channel ->
  ?minx:float ->
  ?maxx:float -> ?miny:float -> ?maxy:float -> ?full_ascii:bool -> unit -> unit
```

Idem, mais avec une liste d'entiers.

Avec support de la couleur du module ANSITerminal.

```
val plotC_array :
  vect:float array ->
  ?title:string ->
  ?param:bool ->
  ?minx:float ->
  ?maxx:float -> ?miny:float -> ?maxy:float -> ?full_ascii:bool -> unit -> unit
```

Affiche les valeurs d'un tableau.

Pour afficher des fonctions.

```
val array_of_fun :
  ?minx:float -> ?maxx:float -> ?nb:int -> f:(float -> 'a) -> 'a array
```

Donne un tableau de valeurs pour une fonction float → float.

```
val array_int_of_fun :
  ?minx:float -> ?maxx:float -> ?nb:int -> f:(int -> 'a) -> 'a array
```

Donne un tableau de valeurs pour une fonction int → int.

```
val plot_fun :
  ?title:string ->
  ?param:bool ->
  ?mon_out:Pervasives.out_channel ->
  ?minx:float ->
  ?maxx:float ->
  ?miny:float ->
  ?maxy:float ->
  ?nb:int -> ?full_ascii:bool -> f:(float -> float) -> unit -> unit
```

Dessine le graphique d'une fonction float → float.

```
val plotC_fun :
  ?title:string ->
  ?param:bool ->
  ?minx:float ->
  ?maxx:float ->
  ?miny:float ->
  ?maxy:float ->
  ?nb:int -> ?full_ascii:bool -> f:(float -> float) -> unit -> unit
```

Dessine le graphique d'une fonction float → float, avec couleurs ANSI.

```
val plot_fun_int :
  ?title:string ->
  ?param:bool ->
  ?mon_out:Pervasives.out_channel ->
  ?minx:float ->
  ?maxx:float ->
  ?miny:float ->
  ?maxy:float ->
  ?nb:int -> ?full_ascii:bool -> f:(int -> float) -> unit -> unit
```

Dessine le graphique d'une fonction float → float.

```
val plotC_fun_int :
  ?title:string ->
  ?param:bool ->
  ?minx:float ->
```

```
  ?maxx:float ->
  ?miny:float ->
  ?maxy:float ->
  ?nb:int -> ?full_ascii:bool -> f:(int -> float) -> unit -> unit
```

Dessine le graphique d'une fonction float → float, avec couleurs ANSI.

Avec un support de la couleur customisable.

```
val plotCc_array :
  vect:float array ->
  ?style_fond:ANSITerminal.style list ->
  ?style_axe:ANSITerminal.style list ->
  ?style_pt:ANSITerminal.style list ->
  ?full_ascii:bool ->
  ?title:string ->
  ?param:bool ->
  ?minx:float -> ?maxx:float -> ?miny:float -> ?maxy:float -> unit -> unit
```

Affiche les valeurs d'un tableau.

Styles ANSI

Voici un rappel des styles disponibles dans le module ANSITerminal : type color = ANSITerminal.color = Black |Red |Green |Yellow |Blue |Magenta |Cyan |White |Default type style = ANSITerminal.style = Reset |Bold |Underlined |Blink |Inverse |Hidden |Foreground of color |Background of color

```
val plotCc_fun :
  f:(float -> float) ->
  ?style_fond:ANSITerminal.style list ->
  ?style_axe:ANSITerminal.style list ->
  ?style_pt:ANSITerminal.style list ->
  ?full_ascii:bool ->
  ?title:string ->
  ?param:bool ->
  ?minx:float ->
  ?maxx:float -> ?miny:float -> ?maxy:float -> ?nb:int -> unit -> unit
```

Dessine le graphique d'une fonction float → float, avec couleurs ANSI customisables.

```
val plotCc_fun_int :
  f:(int -> float) ->
  ?style_fond:ANSITerminal.style list ->
  ?style_axe:ANSITerminal.style list ->
  ?style_pt:ANSITerminal.style list ->
  ?full_ascii:bool ->
  ?title:string ->
  ?param:bool ->
  ?minx:float ->
  ?maxx:float -> ?miny:float -> ?maxy:float -> ?nb:int -> unit -> unit
```

Dessine le graphique d'une fonction float → float, avec couleurs ANSI customisables.

Un essai d'extension de syntaxe

Afin de se rapprocher de la syntaxe de plot en Maple par exemple.

```
val (--) : 'a -> 'b -> 'a * 'b
```

Pour crÃ©er un interval.

```
val plot : ?full_ascii:bool -> (float -> float) * ('a * 'b) -> unit
```

Syntaxe amÃ©liorÃ©e. Essayez : `plot(cos,0-7);;`

Un eval style Maple : trÃ¨s trÃ¨s salement codÃ© en ocaml...

```
val evals : ?p:int -> ?nomv:string -> ?x:'a -> string -> float
```

Permet d'Ã©valuer le contenu d'une string ocaml.

```
val eval : ?nomv:string -> ?x:'a -> 'b -> float
```

Et cette fonction permet d'Ã©valuer un Ã©lÃ©ment ocaml. Si le type de l'Ã©lÃ©ment n'est pas simple, 0.0 est renvoyÃ©.

```
val pplot : ?nomv:string -> ?full_ascii:bool -> 'a -> 'b * 'c -> unit
```

On utilise donc tout ca pour proposer une mÃ©thode de dessin de graphe, fortement polymorphique, mais trÃ¨s trÃ¨s salement codÃ©e (Si vous pensez que j'exagÃ¨re ou si vous pensez le contraire, jetez un oeil attentif a ce qui est fait dans ce programme et dans le module Surcharge : c'est sale !)

Pour le moment, sans vraiment de raison, c'est trÃ¨s lent. En bÃ©ta !

```
val polyPlot :
  ?nomv:string -> ?range:'a * 'b -> ?full_ascii:bool -> 'c -> unit
```

Le mieux que j'ai pu faire. On a ici une fonction qui peut afficher : des constantes entiÃ¨res ou flottantes, des tableaux ou des listes d'entiers ou de flottants, et des chaines de caractÃ¨res reprÃ©sentant des expressions ayant une variable libre. Ca marche trÃ¨s bien ! Mais qu'est-ce que c'est sale... Et c'est lent pour les chaines de caractÃ¨res.

Essayez : polyPlot ~range:(0–1) "acos(x)";; polyPlot ~range:(-7,7) "cos(2*x)";; let a = Array.create 500 0 in for i=0 to 499 do a.(i) ← 7*i+9*i*i+3; done polyPlot a;;

3 Module `MOcamlPlot_noANSI` : Plot : code ml pour ocamlc

Plot : Projet MOcaml

ENS de Cachan - 2012 L3 Informatique Projet MOcaml v1.0.4

Besson Lilian. InspirÃ© de 'meta_plot.c' et 'script_metaplot.sh' Ã©crits par Lilian Besson en FÃ©vrier 2012. (c) Naereen Corp.

```
val line : unit -> int
```

`val column : unit -> int`
RÃ©cupÃ¨re la taille de la fenÃ¨tre ANSI.

`val arrayfloat_of_arrayint : int array -> float array`
Change un tableau d'entiers en tableau de flottants.

`val bool_of_int : int -> bool`

`val int_of_bool : bool -> int`
Conversions entiers et boolÃ©ens.

`val bool_of_float : float -> bool`

`val float_of_bool : bool -> float`
Conversions entiers et flottants.

`val len : 'a array -> int`

`val maxminof_array : ('a -> 'a -> bool) -> 'a array -> 'a`
Calcul du max/min d'un tableau quelconque pour un ordre quelconque.

`val maxof_array : 'a array -> 'a`

`val minof_array : 'a array -> 'a`
Calcul du max.

Calcul du min.

`val (^=) : string Pervasives.ref -> string -> unit`
Actualise une rÃ©fÃ©rence string.

`exception Mauvaise_valeur_gauche`

`exception Mauvaise_valeur_droite`
Si le programme cherche Ã Ã©valuer une valeur trop a gauche (bug).

`exception Dessin_impossible`
Si le programme cherche Ã Ã©valuer une valeur trop a droite (bug).

Si les paramÃ¨tres du tracÃ© font qu'il est impossible a rÃ©aliser. Par exemple si `maxx` < `minx` ou `maxy` < `miny`.

`val getchar : unit -> char`
Comme le getchar du C.
ParamÃªtres graphiques du tracÃ©s.

```
type fleches =
  | Haut
  | Bas
  | HautG
  | BasG
  | Gauche
  | Droite
  | GaucheG
  | DroiteG
  | DiagHautDroite
  | DiagHautGauche
```

```
  | DiagBasDroite
  | DiagBasGauche
  | DiagHautDroiteG
  | DiagHautGaucheG
  | DiagBasDroiteG
  | DiagBasGaucheG
  | AxeHoriz
  | AxeVerti
  | Trou
  | Centre
  | LegX
```

DiffÃ©rentes flÃ¨ches et Ã©lÃ©ment d'un tracÃ©.

```
val fleche : bool -> fleches -> string
```

MÃ©thode pour afficher les flÃ¨ches et les diffÃ©rents Ã©lÃ©ments d'un tracÃ©s. Si vous voulez customisÃ© l'affichage, il suffit de le faire par lÃ . Pas besoin de rentrer dans le code.

Afficher un tableau de flottants.

```
val plot_array :
  vect:float array ->
  ?title:string ->
  ?param:bool ->
  ?mon_out:Pervasives.out_channel ->
  ?minx:float ->
  ?maxx:float -> ?miny:float -> ?maxy:float -> ?full_ascii:bool -> unit -> unit
```

Affiche les valeurs d'un tableau.

```
val plot_list :
  liste:float list ->
  ?title:string ->
  ?param:bool ->
  ?mon_out:Pervasives.out_channel ->
  ?minx:float ->
  ?maxx:float -> ?miny:float -> ?maxy:float -> ?full_ascii:bool -> unit -> unit
```

Idem, mais avec une liste.

```
val plot_array_int :
  vect:int array ->
  ?title:string ->
  ?param:bool ->
  ?mon_out:Pervasives.out_channel ->
  ?minx:float ->
  ?maxx:float -> ?miny:float -> ?maxy:float -> ?full_ascii:bool -> unit -> unit
```

Idem, mais avec un tableau d'entiers.

```
val plot_list_int :
  liste:int list ->
  ?title:string ->
  ?param:bool ->
```

```
  ?mon_out:Pervasives.out_channel ->
  ?minx:float ->
  ?maxx:float -> ?miny:float -> ?maxy:float -> ?full_ascii:bool -> unit -> unit
```
Idem, mais avec une liste d'entiers.
Pour afficher des fonctions.

```
val array_of_fun :
  ?minx:float -> ?maxx:float -> ?nb:int -> f:(float -> 'a) -> 'a array
```
Donne un tableau de valeurs pour une fonction float → float.

```
val array_int_of_fun :
  ?minx:float -> ?maxx:float -> ?nb:int -> f:(int -> 'a) -> 'a array
```
Donne un tableau de valeurs pour une fonction int → int.

```
val plot_fun :
  ?title:string ->
  ?param:bool ->
  ?mon_out:Pervasives.out_channel ->
  ?minx:float ->
  ?maxx:float ->
  ?miny:float ->
  ?maxy:float ->
  ?nb:int -> ?full_ascii:bool -> f:(float -> float) -> unit -> unit
```
Dessine le graphique d'une fonction float → float.

```
val plot_fun_int :
  ?title:string ->
  ?param:bool ->
  ?mon_out:Pervasives.out_channel ->
  ?minx:float ->
  ?maxx:float ->
  ?miny:float ->
  ?maxy:float ->
  ?nb:int -> ?full_ascii:bool -> f:(int -> float) -> unit -> unit
```
Dessine le graphique d'une fonction float → float.
Un essai d'extension de syntaxe
Afin de se rapprocher de la syntaxe de plot en Maple par exemple.

```
val x : float Pervasives.ref
```
A voir.

```
val init : unit -> unit
```
Initialise MOcaml Plot. A complÃ©ter au besoin.
Surcharge des opÃ©rateurs

```
val typage : 'a -> string
```
Fonction de typage

```
val bool_of_int : int -> bool
val int_of_bool : bool -> int
```
Conversions entiers et boolÃ©ens.

`val bool_of_float : float -> bool`

`val float_of_bool : bool -> float`

Conversions entiers et flottants.

`val to_float : 'a -> float`

Converti vers les flottants. Marche bien pour les boolÃ©ens aussi.

`val to_array : 'a -> float array`

Convertit dans un tableau. Marche bien pour les boolÃ©ens aussi.

`val to_array : 'a -> float array`

Convertit dans un tableau. Marche bien pour les boolÃ©ens aussi.

`val (^=) : string Pervasives.ref -> string -> unit`

Actualise une rÃ©fÃ©rence string.

`val to_string : 'a -> string`

Affichage gÃ©nÃ©rique

`val print : 'a -> unit`

`val (--) : 'a -> 'b -> 'a * 'b`

Pour crÃ©er un interval.

`val plot : ?full_ascii:bool -> (float -> float) * ('a * 'b) -> unit`

Syntaxe amÃ©liorÃ©e.

Encore plus d'extensions : vers de la surcharge ?

`val __nom_temporaire__ : string`

`val __nom_temporaire2__ : string`

Fichiers temporaires utilisÃ©s pour la suite.

`val __parse_fichier : string -> string`

Renvoie le contenu de la premiÃ¨re ligne d'un fichier texte. L'argument peut Ãªtre aussi bien l'adresse relative qu'absolue.

`val res : unit -> float`

On va Ã©crire un programme qui Ã©crira le rÃ©sultat de son calcul dans le second fichier temporaire. Cette fonction en rÃ©cupÃ¨re la valeur.

`val execcaml : string -> unit -> unit`

On interprÃ©tera le fichier TEMP avec cette fonction.

`val evals : ?p:int -> ?nomv:string -> ?x:'a -> string -> float`

Un eval style Maple : trÃ¨s trÃ¨s salement codÃ© en ocaml...

Permet d'Ã©valuer le contenu d'une string ocaml.

On utilise le petit module surcharge fait pour le projet.

`val eval : ?nomv:string -> ?x:'a -> 'b -> float`

Et cette fonction permet d'Ã©valuer un Ã©lÃ©ment ocaml. Si le type de l'Ã©lÃ©ment n'est pas simple, 0.0 est renvoyÃ©.

```
val pplot : ?nomv:string -> ?full_ascii:bool -> 'a -> 'b * 'c -> unit
```

On utilise donc tout ca pour proposer une mÃ©thode de dessin de graphe, fortement polymorphique, mais trÃ¨s trÃ¨s salement codÃ©e (Si vous pensez que j'exagÃ¨re ou si vous pensez le contraire, jetez un oeil attentif a ce qui est fait dans ce programme et dans le module Surcharge : c'est sale !)

Pour le moment, sans vraiment de raison, c'est trÃ¨s lent. En bÃ©ta !

```
val polyPlot :
  ?nomv:string -> ?range:'a * 'b -> ?full_ascii:bool -> 'c -> unit
```

Le mieux que j'ai pu faire. On a ici une fonction qui peut afficher : des constantes entiÃ¨res ou flottantes, des tableaux ou des listes d'entiers ou de flottants, et des chaines de caractÃ¨res reprÃ©sentant des expressions ayant une variable libre. Ca marche trÃ¨s bien ! Mais qu'est-ce que c'est sale...