



Étude de la sécurité intrinsèque des langages fonctionnels (LaFoSec)

Titre	Outils associés au langage OCaml
Identifiant	Livable L3.2
Version	3.0
Date	2011-07-07
Pages	33
Approbation	Christèle Faure, SafeRiver
	Date: _____ Signature: _____

Table des révisions

Version	Date	Description et changements	Parties modifiées
1.0	2011-06-16	Version initiale	Tout le document
2.0	2011-06-28	Version contenant les commentaires de l'ANSSI	Tout le document
3.0	2011-07-02	Prise en compte des commentaires de l'ANSSI	Tout le document

Résumé

Ce document, réalisé dans le cadre du projet LaFoSec, propose une description des outils associés au langage OCaml et étudie leur apport à la sûreté et à la sécurité de l'application développée.

Table des matières

Introduction	6
1 Édition	8
1.1 Mode Emacs <code>Cam1</code>	8
1.2 Mode Emacs <code>Tuareg</code>	9
1.3 Extensions pour Eclipse <code>OcaIDE</code> et <code>ODT</code>	11
1.4 Système de documentation <code>ocamldoc</code>	12
2 <i>Parsing et preprocessing</i>	14
2.1 Analyse lexicale avec <code>ocamllex</code> et syntaxique avec <code>ocamlyacc</code> .	14
2.2 <i>Preprocessing</i> avec <code>camlp4</code>	15
3 Compilation	17
3.1 Compilateurs natif <code>ocamlopt</code> et bytecode <code>ocamlc</code>	17
3.2 Moteur de compilation <code>ocamlbuild</code>	18
3.3 Gestionnaire de bibliothèques <code>Findlib</code>	19
4 Observation de codes source	20
4.1 Extracteur de dépendances <code>ocamldep</code>	20
4.2 Navigateur de bibliothèques <code>ocamlbrowser</code>	21
5 Exécution	24
5.1 Machine virtuelle <code>ocamlrun</code>	24
5.2 Boucle interactive <code>ocaml</code> et <code>ocamlmktop</code>	24
6 Observation d'exécution	26
6.1 Débogueur <code>ocamldebug</code>	26
6.2 Profilage par compteurs <code>ocamlprof</code>	27
6.3 Profilage du temps de calcul par <code>gprof</code>	29

7	Liste des licences	30
	Bibliographie	31
	Table des figures	31
	Acronymes	33

Introduction

Objet du document

Ce document a été produit dans le cadre de l'étude LaFoSec, relative au marché n° 2010027960021207501 notifié le 8 novembre 2010 par le Secrétariat Général de la Défense et de la Sécurité Nationale (SGDSN). Il présente les outils associés au langage OCaml.

Présentation du projet LaFoSec

Les langages de programmation dits *fonctionnels* sont réputés offrir de nombreuses garanties facilitant le développement de logiciels soumis à des exigences de sûreté ou de sécurité. Par exemple, la société *Ericsson* a développé le langage fonctionnel Erlang, dédié à la concurrence, pour ses applications de communication. La société *Esterel Technologies* a développé le langage fonctionnel SCADE, dédié au traitement synchrone de flots de données, pour le traitement de logiciel critique.

Dans le cadre de ses activités d'expertise, l'ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information) souhaite bénéficier d'une assistance scientifique et technique sur l'adéquation de ces langages au développement d'applications de sécurité et disposer d'une étude permettant d'améliorer la confiance vis-à-vis de la sécurité de ces applications. C'est l'objet du projet LaFoSec.

Le projet LaFoSec consiste en une étude prospective des langages fonctionnels, visant à déterminer les caractéristiques propres à ces langages susceptibles de répondre aux exigences de sécurité.

Tout d'abord, les caractéristiques des langages fonctionnels sont décrites et plusieurs langages généralistes sont décrits selon ces caractéristiques. Ensuite, cette étude est approfondie en se restreignant à trois langages, OCaml, F# et Scala, pour recenser leurs avantages et inconvénients du point de vue de la sécurité.

Présentation de l'objet du document

Ce document décrit succinctement les outils couramment utilisés pour le développement d'applications en OCaml et identifie pour chacun les apports à la sûreté et à la sécurité de l'application développée. Ils sont décrits dans d'autres documents fournis par l'étude LaFoSec : [ANA-SÉCU, 2011], [MODE-EX, 2011] et [RECOM-OCAML, 2011].

Les outils présentés sont documentés et pérennes car le plus souvent inclus dans la distribution officielle de l'INRIA. Ces outils traitent l'intégralité du langage OCaml en conformité avec sa sémantique. Il existe d'autres outils non présentés dans ce document. Ils sont référencés sur le site officiel de OCaml via la Caml Hump¹.

Pour la plupart, ces outils sont distribués sous forme de logiciel libre sous licence QPL avec exception (diminuant les obligations pour un usage strictement privé). Toutes les licences du monde libre citées dans ce document sont groupées dans un tableau section 7. Les membres du consortium industriel OCaml² sont autorisés à redistribuer ces outils sous la licence de leur choix.

1. <http://caml.inria.fr/cgi-bin/hump.fr.cgi>

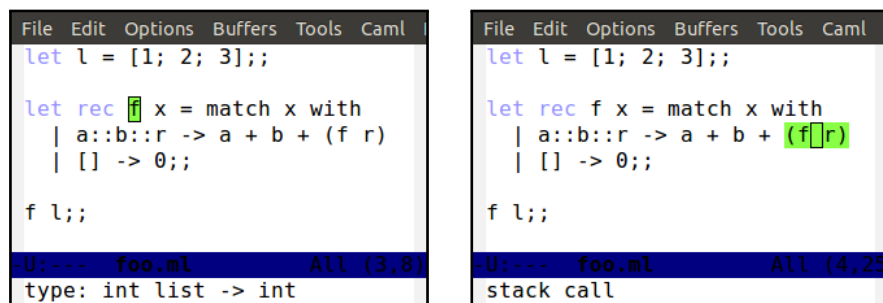
2. <http://caml.inria.fr/consortium/>

Chapitre 1

Édition

1.1 Mode Emacs Caml

Description. La distribution OCaml contient un mode pour l'éditeur Emacs. Ce mode offre de nombreuses fonctionnalités telles que la coloration de code et l'indentation automatique. Il s'interface avec la boucle interactive `ocaml` (cf. section 5.2), le débogueur `ocamldebug` (cf. section 6.1) et le compilateur (cf. section 3.1) en localisant directement dans le source les erreurs détectées. En utilisant les informations fournies par l'option `-annot` du compilateur (cf. section 3.1), il est possible d'afficher interactivement le type d'une expression à un point donné du code source, la portée d'un identifiant et la nature d'un appel de fonction (terminal ou non).



```
File Edit Options Buffers Tools Caml |
let l = [1; 2; 3];;

let rec f x = match x with
| a::b::r -> a + b + (f r)
| [] -> 0;;

f l;;

U:-- foo.ml All (3,8)
type: int list -> int
```

```
File Edit Options Buffers Tools Caml |
let l = [1; 2; 3];;

let rec f x = match x with
| a::b::r -> a + b + (f r)
| [] -> 0;;

f l;;

U:-- foo.ml All (4,25)
stack call
```

FIGURE 1.1 – Mode Caml : visualisation d'informations (type et nature d'appel)

La fonction `f` n'est pas exhaustive car il manque les cas où la liste contient un ou deux éléments. Cet exemple est utilisé tout au long de ce document pour exhiber les capacités des différents outils.


```

File Edit Options Buffers Tools Caml |
let l = [1; 2; 3];;

let rec f x = match x with
| a::b::r -> a + b + (f r)
| [] -> 0;;

f l;;

U:--- foo.ml All (4,19)
b is bound at line 4 char 7

```

```

File Edit Options Buffers Tools Caml |
let l = [1; 2; 3];;

let rec f x = match x with
| a::b::r -> a + b + (f r)
| [] -> 0;;

f l;;

U:--- foo.ml All (3,10)
local variable x is bound here

```

FIGURE 1.2 – Mode Caml : visualisation d'inférences (liaison et portée)

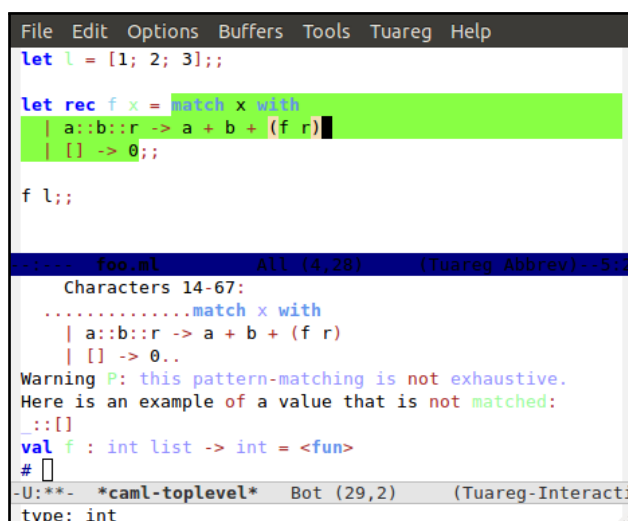
Apports pour la sûreté ou la sécurité. Le mode Caml intègre les différents outils OCaml à l'éditeur Emacs. Il n'apporte pas de garanties supplémentaires à celles de ces outils.

Compatibilité avec le langage. Le mode Caml est compatible avec l'ensemble du langage OCaml, et est régulièrement mis à jour pour prendre en compte ses évolutions. Mais des différences entre l'interprétation OCaml et la coloration peuvent apparaître sur des points particuliers tels que les commentaires en présence des caractères " et '. Dans ces cas particuliers, la coloration peut donc être trompeuse. Il est largement utilisé par la communauté des développeurs OCaml.

Licence et distribution. Le mode Caml est disponible sous licence GPL (voir 7). Il est inclus dans la distribution OCaml.

1.2 Mode Emacs Tuareg

Description. Tuareg est un mode OCaml pour l'éditeur Emacs qui constitue une alternative au mode Emacs Caml inclus dans la distribution OCaml. Il effectue l'indentation automatique et la coloration syntaxique du code. Il permet d'évaluer le code dans une boucle interactive (cf. section 5.2) et de déboguer en interfaçant `ocamldebug` (cf. section 6.1). Il rend possible la complétion des identifiants, la recherche de la documentation qui leur est associée, et, si le fichier est compilé avec l'option `-annot`, l'affichage du type d'une sous-expression à un point donné (cf. section 3.1).



```
File Edit Options Buffers Tools Tuareg Help
let l = [1; 2; 3];;

let rec f x = match x with
| a::b::r -> a + b + (f r)
| [] -> 0;;

f l;;

----- top.ml ----- All (4,28) ----- (Tuareg Abbrev)-----502
Characters 14-67:
.....match x with
| a::b::r -> a + b + (f r)
| [] -> 0..
Warning P: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
::[]
val f : int list -> int = <fun>
#
-U:***- *caml-toplevel* Bot (29,2) ----- (Tuareg-Interacti
type: int
```

FIGURE 1.3 – Mode **Tuareg** : boucle interactive et visualisation d'inférences (type)

Apports pour la sûreté ou la sécurité. Le mode **Tuareg** intègre les différents outils OCaml à l'éditeur Emacs. Il n'apporte pas de garanties supplémentaires à celles de ces outils.

Compatibilité avec le langage. Le mode **Tuareg** est compatible avec l'ensemble du langage OCaml, et est régulièrement mis à jour pour prendre en compte ses évolutions. Mais des différences entre l'interprétation OCaml et la coloration peuvent apparaître sur des points particuliers tels que les commentaires en présence des caractères " et '. Dans ces cas particuliers, la coloration peut donc être trompeuse. Il est largement utilisé par la communauté des développeurs OCaml.

Licence et distribution. Le mode **Tuareg** est disponible sous licence GPL (voir 7). Il est maintenu par *Jane Street*. On peut le récupérer depuis le site OCaml Forge¹, et il est présent dans la plupart des distributions Linux.

1. <https://forge.ocamlcore.org/projects/tuareg/>

1.3 Extensions pour Eclipse OcaIDE et ODT

Description. L'environnement de développement intégré (IDE) Eclipse propose un mécanisme d'extensions. Il existe plusieurs extensions OCaml pour Eclipse. Ainsi les extensions OcaIDE² et ODT (OCaml Development Tools)³ permettent une gestion de projet OCaml intégrée dans l'environnement Eclipse.

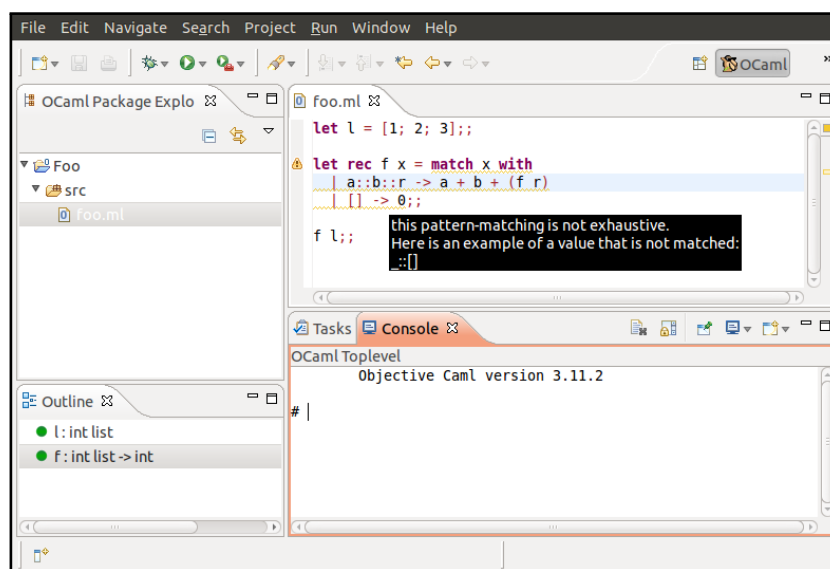


FIGURE 1.4 – Environnement de développement ODT pour Eclipse

Apports pour la sûreté ou la sécurité. De la même manière que les modes Emacs pour OCaml, ces extensions intègrent les différents outils OCaml à l'environnement Eclipse. Ils n'apportent pas de garanties supplémentaires à celles de ces outils.

Compatibilité avec le langage. Ces extensions sont compatibles avec l'ensemble du langage OCaml. Mais des différences entre l'interprétation OCaml et la coloration peuvent apparaître sur des points particuliers tels que les commentaires en présence des caractères " et '. Dans ces cas particuliers, la coloration peut donc être trompeuse.

2. <http://www.algo-prog.info/ocaide/>

3. <http://ocamltdt.free.fr/>

Licence et distribution. L'extension `OcaIDE` est disponible sous licence CeCILL-B (voir 7), l'extension `ODT` sous LGPL (voir 7). Ces deux extensions sont distribuées par la communauté de développeurs OCaml, leur installation est facilitée par le mécanisme d'installation de Eclipse.

1.4 Système de documentation ocaml doc

Description. L'outil `ocaml doc` est un système de documentation pour OCaml. Des commentaires structurés, de la forme `(** ... *)`, permettent de placer dans le code source des descriptions des différents éléments composant le code. À partir de ces commentaires, des fichiers d'implémentation (fichiers `.ml`) et des fichiers d'interface (fichiers `.mli`), il génère une documentation au format HTML, \LaTeX , \TeX info, ou `man`. Un langage simple de balises permet une mise en forme des commentaires. Des étiquettes permettent d'associer une signification particulière à certains éléments des commentaires. Une documentation inclut le type de chacun des éléments du code source. `ocaml doc` fait appel au compilateur OCaml pour inférer et vérifier ces informations de typage. Il se base pour cela sur les fichiers d'interface compilés (fichier `.cmi`).

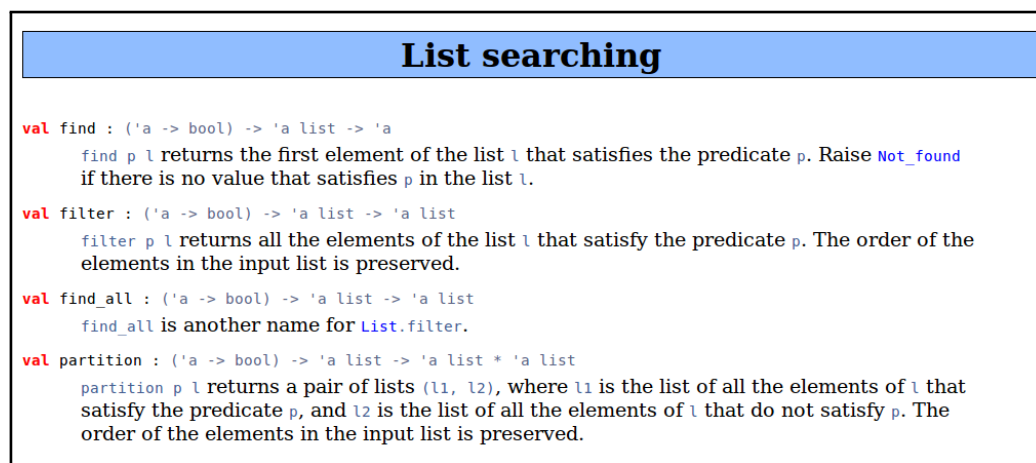


FIGURE 1.5 – Documentation de la bibliothèque standard produite par `ocaml doc`

Apports pour la sûreté ou la sécurité. L'outil `ocaml doc` facilite la documentation de code en fournissant un mécanisme intégré au processus de compilation, garantissant ainsi la pleine concordance entre le code exécuté et la

documentation.

Compatibilité avec le langage. Le générateur de documentation `ocamldoc` couvre l'ensemble des constructions du langage. Ses fonctionnalités sont documentées dans le chapitre 15 du manuel de référence [Leroy *et al.*, 2010].

Licence et distribution. La licence de `ocamldoc` est la QPL avec exception. Cet outil est inclus dans la distribution OCaml.

Chapitre 2

Parsing et preprocessing

2.1 Analyse lexicale avec `ocamllex` et syntaxique avec `ocamlyacc`

Description. L'outil `ocamllex` est un générateur de lexers (analyseurs lexicaux). Il est l'équivalent pour OCaml de `lex` pour le langage C.

Il génère un lexer OCaml (fichier OCaml) à partir d'un ensemble de règles qui associent des actions (code OCaml) à des expressions régulières. Ces règles sont décrites dans un fichier `.mll`.

Le lexer généré définit pour chaque règle lexicale une fonction prenant parmi ses arguments un tampon de type `Lexing.lexbuf`.

L'outil `ocamlyacc` est un générateur de parseurs (analyseurs syntaxiques). Il est l'équivalent pour OCaml de `yacc` du langage C.

Il génère un parseur OCaml à partir d'une grammaire non contextuelle et des actions (code OCaml) associées aux entrées de la grammaire. À partir de cette grammaire décrite dans un fichier `.mly`, `ocamlyacc` produit un fichier `.ml` et un fichier `.mli`.

Le parseur généré définit pour chaque entrée de la grammaire une fonction prenant comme argument un lexer du même type que ceux générés par `ocamllex` et un tampon de type `Lexing.lexbuf`.

Apports pour la sûreté ou la sécurité. Les outils d'analyses lexicales et syntaxiques permettent d'implémenter facilement la validation des données complexes en entrée de programmes (le livrable [RECOM-OCAML, 2011] en recommande l'usage en lieu et place des fonctionnalités du module `Marshal`).

Compatibilité avec le langage. Les outils `ocamllex` et `ocamlyacc` sont compatibles avec les autres constructions du langage OCaml et produisent du code source OCaml. Les fonctionnalités de `ocamllex` et `ocamlyacc` sont documentées dans le chapitre 12 du manuel de référence [Leroy *et al.*, 2010].

Licence et distribution. La licence de `ocamllex` et de `ocamlyacc` est la QPL avec exception. Ces outils sont inclus dans la distribution OCaml.

2.2 *Preprocessing* avec camlp4

Description. L'outil `camlp4` est un préprocesseur et un *pretty printer* pour OCaml. Il permet de générer des analyseurs syntaxiques et des *printers* extensibles. Le parseur généré par `camlp4` peut être utilisé par le compilateur à la place du parseur OCaml, grâce à l'utilisation de l'option `-pp`. Il produit un AST OCaml garanti bien formé. Celui-ci est transmis au compilateur qui effectue les passes habituelles de typage et de production de code.

L'outil `camlp4` est particulièrement adapté pour les tâches de *preprocessing* pour OCaml grâce aux fonctionnalités suivantes :

- L'ajout ou le retrait de constructions syntaxiques au parseur OCaml extensible de `camlp4` permet d'étendre ou de restreindre le syntaxe concrète du langage OCaml sans modifier le compilateur lui-même.
- Des extensions localisées entre doubles chevrons (*quotes*) `<< ... >>` permettent de définir des syntaxes concrètes et leurs parseurs pour faciliter la manipulation de certaines données. Le parseur OCaml généré par `camlp4` rencontrant des chevrons appelle le parseur dédié.

Les parseurs et *printers* `camlp4` sont écrits en OCaml et offrent donc les mêmes garanties que tout programme OCaml.

Apports pour la sûreté ou la sécurité. Voir section 2.1. Les documents [ANA-SÉCU, 2011] et [MODE-EX, 2011] indiquent qu'il est possible d'utiliser `camlp4` avec l'option `-pp` du compilateur OCaml pour restreindre la syntaxe du langage en interdisant certaines constructions dangereuses. Tout parseur généré par `camlp4` peut aussi être utilisé indépendamment du compilateur pour effectuer une analyse de conformité de la syntaxe associée.

Par exemple, l'extension `camlp4` suivante permet de lever une exception au moment de la compilation d'un fichier source OCaml si celui-ci contient un appel

au module `Obj`¹.

```
EXTEND Gram

  GLOBAL: a_UIDENT;

  a_UIDENT:
    [ [ 'UIDENT "Obj" -> raise Use_of_Obj
      | 'UIDENT s -> s
      ] ];

END;
```

FIGURE 2.1 – Extension `camlp4` interdisant l'appel au module `Obj`

Compatibilité avec le langage. L'outil `camlp4` est compatible avec l'ensemble du langage OCaml.

Le manuel de référence OCaml ne fait malheureusement pas mention de `camlp4`. Une documentation incomplète est disponible en ligne².

Il est à noter qu'une autre version de `camlp4` est actuellement maintenue hors de la distribution officielle sous le nom de `camlp5`³. Bien que ces deux outils portent des noms similaires et implémentent les mêmes fonctionnalités, ils ne sont pas compatibles.

Licence et distribution. La licence de `camlp4` est la LGPL avec exception (voir 7). Cet outil est inclus dans la distribution OCaml.

1. Le module `Obj` est considéré non-sûr, cf. [ANA-SÉCU, 2011] pour plus de détails.
2. <http://brion.inria.fr/gallium/index.php/Camlp4>
3. <http://pauillac.inria.fr/~ddr/camlp5/>

Chapitre 3

Compilation

3.1 Compilateurs natif `ocamlopt` et bytecode `ocamlc`

Description. La distribution OCaml contient deux spécialisations de son compilateur produisant pour l'un, `ocamlopt`, du code natif et pour l'autre, `ocamlc`, du bytecode. Le code natif est directement exécutable sur l'architecture ciblée, le bytecode est lui exécutable par la machine virtuelle OCaml (cf. section 5.2).

Le compilateur OCaml propose de nombreuses options de compilation décrites dans les chapitres 8 et 11 du manuel de référence [Leroy *et al.*, 2010]. Ne sont mentionnées ici que les deux options permettant d'outiller le développement.

L'option `-i` génère un fichier d'interface `.mli` contenant les prototypes (noms et types inférés et vérifiés) des éléments définis dans un fichier d'implémentation `.ml` donné.

L'option `-annot` génère, pour un fichier d'implémentation donné, un fichier d'annotations (d'extension `.annot`) contenant les types de toutes les sous-expressions, les informations de portée de chaque identifiant et la nature des appels des fonctions (terminal ou non). Les environnements de développement de OCaml utilisent généralement ces informations pour fournir des informations contextuelles au programmeur (cf. les modes pour Emacs en sections 1.1 et 1.2). Ce fichier d'annotations est aussi généré en cas d'erreur de compilation et permet d'afficher les types inférés avant détection de l'erreur.

Apports pour la sûreté ou la sécurité. La sûreté de fonctionnement induite par les vérifications de typage et de couverture de filtrage est mise en œuvre par le

compilateur OCaml. Le compilateur OCaml est analysé du point de vue de la sécurité dans les livrables [ANA-SÉCU, 2011] et [MODE-EX, 2011]. Leurs options sont étudiées du point de vue de la sécurité dans les livrables [MODE-EX, 2011] et [RECOM-OCAML, 2011].

Compatibilité avec le langage. Les deux versions du compilateur proposent deux modèles d'exécution différents mais amplement compatibles : un code source aura le même comportement quel que soit le modèle choisi à quelques différences près, décrites dans [MODE-EX, 2011].

Licence et distribution. Les compilateurs `ocamlopt` et `ocamlc` sont distribués sous licence QPL avec exception. Ils sont inclus dans la distribution OCaml.

3.2 Moteur de compilation `ocamlbuild`

Description. L'outil `ocamlbuild` est un moteur de compilation dédié à OCaml. Il facilite la compilation de projets de développement en automatisant l'analyse des dépendances entre fichiers (en faisant appel à `ocamldep`, voir section 4.1), en automatisant les commandes de compilation selon les besoins (appels aux préprocesseurs, lexers, parseurs et compilateurs avec les options `-c` puis `-o`) et en plaçant les fichiers résultant de la compilation dans un répertoire séparé. La compilation avec `ocamlbuild` ne requiert généralement qu'une seule ligne de commande et n'affiche que les informations essentielles au suivi du processus de compilation. `ocamlbuild` possède un grand nombre de règles de compilation : en particulier en mode natif, bytecode, débogue et profilage. À la manière d'un `Makefile`, il permet de définir des directives personnalisées.

Apports pour la sûreté ou la sécurité. L'outil `ocamlbuild` apporte une simplification et une accélération du processus de compilation. Il n'apporte pas de garanties particulières en terme de sûreté ou de sécurité.

Compatibilité avec le langage. Le moteur de compilation `ocamlbuild` couvre la plupart des outils de la distribution OCaml et de ce fait l'ensemble du langage. Le manuel de référence OCaml ne fait malheureusement pas mention de `ocamlbuild`, seule une page `man` est incluse dans la distribution OCaml.

Une documentation est toutefois disponible en ligne¹. Une extension appelée `ocamlbuild-plus` ajoute une compatibilité avec Findlib (voir section 3.3).

Licence et distribution. La licence de `ocamlbuild` est la QPL avec exception. Ce programme est inclus dans la distribution OCaml.

3.3 Gestionnaire de bibliothèques Findlib

Description. Le gestionnaire de bibliothèques Findlib fournit une commande `ocamlfind` enveloppant les commandes `ocamlc` et `ocamlopt` (voir section 3.1), `ocamlmktop` (voir section 5.2), `ocamldep` (voir section 4.1), `ocamlDOC` (voir section 1.4), et `ocamlcp` (voir section 6.2). Findlib propose une méthode de gestion de bibliothèques OCaml sous forme de paquetages (*packages*). Aux sources d'un paquetage est associé un fichier indiquant la version, les options de compilation et les dépendances du paquetage dans un fichier appelé `META` sous un format dédié. La commande `ocamlfind` permet de consulter et de modifier la base locale de paquetages à l'aide des sous-commandes : `query`, `install`, `remove`, `printconf` et `list`.

Apports pour la sûreté ou la sécurité.

L'utilisation systématique d'un gestionnaire de bibliothèques tel que Findlib offre une meilleure traçabilité des unités de compilation. En particulier cet outil permet d'identifier les fichiers et bibliothèques appartenant à l'application et donc de contrôler strictement leur origine.

Compatibilité avec le langage. Même si Findlib ne fait pas partie de la distribution OCaml, il est très largement utilisé par la communauté d'utilisateurs OCaml. La distribution de paquetages `GODI`² ou les paquetages de la distribution Linux/Debian en font usage.

Licence et distribution. Findlib³ est disponible sous licence MIT/X11 (voir 7). Il est distribué sous forme de paquetage pour la plupart des distributions Linux.

1. <http://nicolaspouillard.fr/ocamlbuild/ocamlbuild-user-guide.pdf>

2. <http://godi.camlcity.org/godi/>

3. <http://projects.camlcity.org/projects/findlib.html>

Chapitre 4

Observation de codes source

4.1 Extracteur de dépendances `ocamldep`

Description. L'outil `ocamldep` génère un fichier de dépendances entre fichiers objet OCaml (`.cmo` et `.cmi`) à partir des fichiers sources (`.ml` et `.mli`). Il calcule une sur-approximation des dépendances qui peut inclure des dépendances inutiles c'est à dire des fichiers qui ne sont pas nécessaires à la compilation. Cette sur-approximation vient du fait qu'il cherche les références aux unités de compilation externes par analyse syntaxique des fichiers sources et non par analyse sémantique.

Le format des dépendances est celui de `make` et peut donc être directement inclus dans un `Makefile`. Des outils comme `ocaml-dot` et `Graphviz` permettent de visualiser ces dépendances sous forme d'un graphe comme illustré ici par le graphe de dépendances de la bibliothèque standard OCaml.

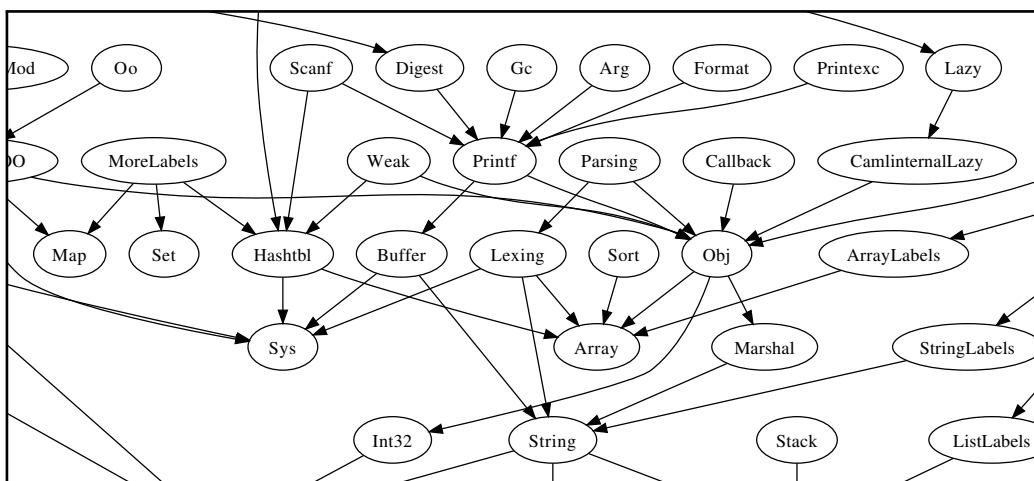


FIGURE 4.1 – Extrait du graphe de dépendances de la bibliothèque standard, produit par `ocamldep` et visualisé à l'aide de `ocamlviz` et `Graphviz`

Apports pour la sûreté ou la sécurité. L'outil `ocamldep` facilite l'analyse de code en produisant une liste des dépendances entre fichiers. Par exemple la commande `ocamldep -modules *.ml | grep Obj` permet de trouver toutes les utilisations du module `Obj` dans un programme.

Compatibilité avec le langage. L'implémentation de `ocamldep` partage son parseur avec celui du compilateur OCaml garantissant ainsi une homogénéité dans leurs analyses syntaxiques.

Licence et distribution. La licence de `ocamldep` est la QPL avec exception. Cet outil est inclus dans la distribution OCaml.

4.2 Navigateur de bibliothèques `ocamlbrowser`

Description. L'outil `ocamlbrowser` est un navigateur de code source OCaml. Il permet d'examiner les valeurs, types, classes, signatures et modules contenus dans les bibliothèques OCaml disponibles dans un environnement donné. La visualisation de la hiérarchie des modules se fait par colonne. Sélectionner un module d'une colonne fait apparaître son contenu dans la colonne de droite. L'interface

(nom et type) de l'élément sélectionné s'affiche dans le cadre du bas. L'outil intègre une méthode de recherche de valeurs de modules par nom et par type.

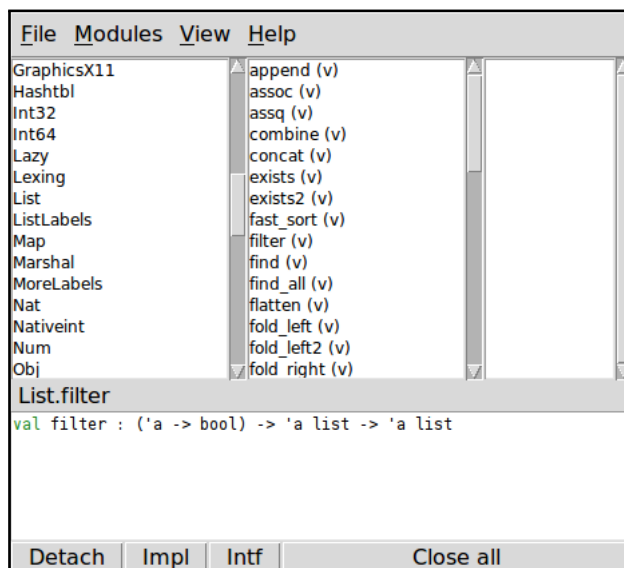


FIGURE 4.2 – Navigation dans la bibliothèque standard avec `ocamlbrowser`

Apports pour la sûreté ou la sécurité. L'outil `ocamlbrowser` facilite la revue manuelle du code en fournissant un affichage des interfaces compilées `.cmi`.

Compatibilité avec le langage. La navigation couvre le contenu des modules et sous-modules, leurs types, valeurs et classes mais ne fournit pas le contenu des signatures (`module type`) ou des foncteurs de modules. De la même manière, la recherche ne couvre pas les signatures, foncteurs, constructeurs de type, champs d'enregistrement et méthodes de classe. D'autre part, l'outil ne permet pas d'inclure les commentaires structurés dans la navigation ou la recherche.

Les informations sur lesquels se base `ocamlbrowser` sont celles des fichiers objets d'interface `.cmi`, ce qui assure que les informations présentées sont correctes vis-à-vis du typage car produites par le compilateur OCaml. Si les fichiers d'interface `.mli` et d'implémentation `.ml` sont aussi disponibles, il y donne un accès direct.

Licence et distribution. La licence de `ocamlbrowser` est la LGPL avec exception de linking (voir 7). Cet outil est inclus dans la distribution OCaml.

Un outil alternatif, `ocamlspotter`, est développé par la communauté d'utilisateurs. Il propose une navigation plus fine dans les sources OCaml et s'intègre à l'éditeur Emacs. Il propose aussi les fonctionnalités associées à l'option `-annot` du compilateur (voir section 3.1).

Chapitre 5

Exécution

5.1 Machine virtuelle `ocamlrun`

Description. Les programmes OCaml compilés avec le compilateur bytecode `ocamlc` (voir section 3.1) sont exécutés par la machine virtuelle `ocamlrun` de OCaml aussi appelée système *run-time* ou interpréteur de bytecode. Ce modèle d'exécution permet au bytecode d'un programme d'être exécuté indifféremment sur différentes plateformes possédant une machine virtuelle OCaml.

Apports pour la sûreté ou la sécurité. La machine virtuelle `ocamlrun` est analysée du point de vue de la sécurité dans les livrables [ANA-SÉCU, 2011] et [MODE-EX, 2011].

Compatibilité avec le langage. Comme indiqué en section 3.1, les modèles d'exécution sont amplement compatibles entre eux.

Licence et distribution. La machine virtuelle `ocamlrun` est distribuée sous licence QPL avec exception. Elle est incluse dans la distribution OCaml.

5.2 Boucle interactive `ocaml` et `ocamlmktop`

Description. L'outil `ocaml` est une boucle interactive pour le langage OCaml offrant une modèle d'exécution Read-Eval-Print Loop (REPL).

Un générateur de boucles interactives `ocamlmktop` permet par ailleurs de créer des versions spécialisées de la boucle interactive. Une boucle interactive peut ainsi avoir des bibliothèques OCaml ou C pré-chargées.

Une boucle interactive réalise une exécution pas-à-pas et permet d'obtenir un retour du typage. Une boucle interactive est particulièrement utile durant la phase de débogue d'un programme. Elle est aussi un outil adaptée à l'apprentissage du langage.

Apports pour la sûreté ou la sécurité. La boucle interactive facilite le test en phase de développement. `ocamlmktop` permet de créer des boucles interactives intégrant des bibliothèques supplémentaires. Ceci permet aux membres d'une équipe de développement d'utiliser systématiquement les mêmes bibliothèques. L'utilisation de toute boucle interactive est déconseillée après cette phase.

Compatibilité avec le langage. Comme indiqué en section 3.1, les modèles d'exécution sont amplement compatibles entre eux.

Licence et distribution. La licence de la boucle interactive `ocaml` et de `ocamlmktop` est la QPL avec exception. Ces outils sont inclus dans la distribution OCaml.

Chapitre 6

Observation d'exécution

6.1 Débogueur `ocamldebug`

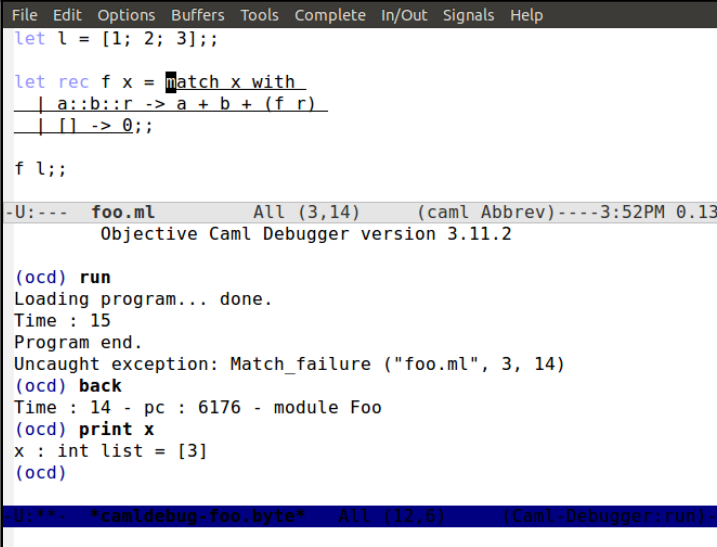
Description. L'outil `ocamldebug` est un débogueur pour OCaml. Il propose les fonctionnalités standard d'un débogueur : exécution pas-à-pas, points d'arrêt et accès aux valeurs. De plus, il permet une exécution en arrière dans le temps. Il est possible de personnaliser `ocamldebug` en programmant ses propres *printers* de valeurs. Le programme à déboguer doit être compilé avec le compilateur bytecode `ocamlc` et l'option `-g`. Ce débogueur s'interface avec les modes `Cam1` et `Tuareg` pour l'éditeur Emacs (voir section 1.1 et 1.2).

Apports pour la sûreté ou la sécurité. L'outil `ocamldebug` fournit un outil d'analyse d'exécution de programme. Il permet une observation fine du comportement du programme développé.

Compatibilité avec le langage. Le débogueur `ocamldebug` couvre l'ensemble des constructions du langage. Ses fonctionnalités sont documentées dans le chapitre 16 du manuel de référence [Leroy *et al.*, 2010].

Le débogueur est disponible sous presque toutes les architectures de la distribution OCaml : sous Windows, le débogueur ne fonctionne qu'avec Cygwin.

Licence et distribution. La licence de `ocamldep` est la QPL avec exception. Il est inclus dans la distribution OCaml.



```
File Edit Options Buffers Tools Complete In/Out Signals Help
let l = [1; 2; 3];;

let rec f x = match x with
| a::b::r -> a + b + (f r)
| [] -> 0;;

f l;;

-U:-- foo.ml All (3,14) (caml Abbrev)---3:52PM 0.13-
Objective Caml Debugger version 3.11.2

(ocd) run
Loading program... done.
Time : 15
Program end.
Uncaught exception: Match_failure ("foo.ml", 3, 14)
(ocd) back
Time : 14 - pc : 6176 - module Foo
(ocd) print x
x : int list = [3]
(ocd)

U:** *camldebug-foo.byte* All (12,6) (Caml Debugger:run)-
```

FIGURE 6.1 – Débogue avec `ocamldebug` intégré au mode `Cam1` de Emacs

6.2 Profilage par compteurs `ocamlprof`

Description. L'outil `ocamlprof` permet d'effectuer le profilage d'un programme. Une version modifiée du compilateur bytecode OCaml, `ocamlcp`, instrumente un programme pour y insérer des compteurs d'appels. Après exécution, `ocamlprof` imprime les sources annotées par les valeurs des compteurs (`(* n *)` pour `n` appels).

```

File Edit Options Buffers Tools Caml Help
let l = [1; 2; 3];;
[]
let rec f x = match x with
| a::b::r -> a + b + (f r)
| [] -> 0;;

f l;;

-U:--- foo.ml All (2,0) (caml Abbrev)----4:07PM-----
let l = [1; 2; 3];;

let rec f x = (* 2 *) match x with
| a::b::r -> (* 1 *) a + b + (f r)
| [] -> (* 0 *) 0;;

f l;;

```

FIGURE 6.2 – Exemple de fichier source OCaml avant et après profilage avec `ocamlprof`

La fonction `f` n'est pas exhaustive car il manque les cas où la liste contient un ou deux éléments. Le profilage montre par les valeurs calculées que l'exécution s'arrête au deuxième appel récursif de `f`.

Apports pour la sûreté ou la sécurité. L'outil `ocamlprof` fournit un outil d'analyse d'exécution de programmes. Il permet par exemple de vérifier la couverture de code d'un jeu de tests en repérant les expressions non exécutées lors du test.

Compatibilité avec le langage. L'outil `ocamlprof` couvre l'ensemble des constructions du langage. Ses fonctionnalités sont documentées dans le chapitre 17 du manuel de référence [Leroy *et al.*, 2010].

Licence et distribution. La licence de `ocamlprof` est la QPL avec exception. Il est inclus dans la distribution OCaml.

6.3 Profilage du temps de calcul par gprof

Description. Il est possible d'interfacer un programme développé en OCaml avec l'outil de profilage `gprof` du projet GNU. Il permet de compter le nombre d'appels et le temps passé à exécuter les instructions natives (du programme et de l'exécutif OCaml).

Il faut pour cela compiler le programme avec le compilateur natif `ocamlc` et l'option de compilation `-p`.

Apports pour la sûreté ou la sécurité. Le profilage du temps de calcul n'apporte pas directement de garanties pour la sûreté ou la sécurité. Il permet cependant de contrôler les temps d'exécution et donc d'identifier d'éventuelles fuites d'information.

Compatibilité avec le langage. L'outil `gprof` est de bas niveau, il permet un profilage au niveau du code natif produit par le compilateur et non au niveau du code source.

Licence et distribution. L'outil GNU `gprof` est développé par le projet GNU sous licence GPL (voir 7).

Chapitre 7

Liste des licences

Les licences citées dans ce document sont listées ci-dessous.

CeCILL-B <http://www.cecill.info/licences.fr.html>
GPL <http://www.opensource.org/licenses/GPL-2.0>
LGPL <http://www.opensource.org/licenses/LGPL-2.1>
MIT/X11 <http://www.opensource.org/licenses/MIT>
QPL <http://www.opensource.org/licenses/QPL-1.0>

Bibliographie

- [ANA-SÉCU, 2011] Analyse des langages OCaml, Scala et F#. Étude de la sécurité intrinsèque des langages fonctionnels (LaFoSec) L2.2.2, ANSSI (2011). Étude menée par un consortium composé de SafeRiver, CEDRIC, Normation et Oppida.
- [Leroy *et al.*, 2010] LEROY, X., DOLIGEZ, D., FRISCH, A., GARRIGUE, J., RÉMY, D. et VOUILLON, J. (2010). The Objective Caml system release 3.12 – documentation and user's manual. INRIA. Disponible en ligne <http://caml.inria.fr/pub/docs/manual-ocaml/>.
- [MODE-EX, 2011] Modèles d'exécution du langage OCaml. Étude de la sécurité intrinsèque des langages fonctionnels (LaFoSec) L3.1.2, ANSSI (2011). Étude menée par un consortium composé de SafeRiver, CEDRIC, Normation et Oppida.
- [RECOM-OCAML, 2011] Recommandations relatives à l'utilisation du langage OCaml et à l'installation et la configuration des outils associés. Étude de la sécurité intrinsèque des langages fonctionnels (LaFoSec) L3.3.2, ANSSI (2011). Étude menée par un consortium composé de SafeRiver, CEDRIC, Normation et Oppida.

Table des figures

1.1	Mode <code>Cam1</code> : visualisation d'informations (type et nature d'appel)	8
1.2	Mode <code>Cam1</code> : visualisation d'inférences (liaison et portée)	9
1.3	Mode <code>Tuareg</code> : boucle interactive et visualisation d'inférences (type)	10
1.4	Environnement de développement ODT pour Eclipse	11
1.5	Documentation de la bibliothèque standard produite par <code>ocamldoc</code>	12
2.1	Extension <code>camlp4</code> interdisant l'appel au module <code>Obj</code>	16
4.1	Extrait du graphe de dépendances de la bibliothèque standard, produit par <code>ocamldep</code> et visualisé à l'aide de <code>ocamlviz</code> et <code>Graphviz</code>	21
4.2	Navigation dans la bibliothèque standard avec <code>ocamlbrowser</code>	22
6.1	Débogue avec <code>ocamldebug</code> intégré au mode <code>Cam1</code> de Emacs	27
6.2	Exemple de fichier source OCaml avant et après profilage avec <code>ocamlprof</code>	28

Acronymes

ANSSI	Agence Nationale de la Sécurité des Systèmes d'Information
AST	Arbre de Syntaxe Abstraite
CeCILL	Licence CEA CNRS INRIA Logiciel Libre
GPL	GNU General Public License
IDE	Integrated Development Environment
INRIA	Institut National de Recherche en Informatique et en Automatique
LGPL	GNU Lesser General Public License
QPL	Q Public License
REPL	Read-Eval-Print Loop
SGDSN	Secrétariat Général de la Défense et de la Sécurité Nationale