Chapitre 1

Rapports de réunions de PSTL : *interopérabilité entre OCaml et Java*

Béatrice CARRE
Encadrants : Emmanuel Chailloux, Xavier Clerc, Grégoire Henry

Introduction

Lien vers l'énnoncé du projet : lien.

1.1 mardi 21 janvier : Le projet

1.1.1 sujets abordés

Pour bien commencer, et établir un environnement de travail pratique pour toute la durée du projet, l'élaboration d'un site de gestion de version et un forum de conversation nous ont été fortement recommandés.

Une fois ceci fait, nous aurons à installer et nous renseigner sur chacun des outils que nous allons manipuler :

- Le projet OCaml-Java, un backend pour le compilateur binaire de Ocaml produisant du bytecode Java. `http://ocamljava.x9c.fr/preview/`
- Le projet O'Jacaré, un générateur de code à partir d'un IDL, permettant l'interopérabilité entre OCaml et Java. `http://www.pps.univ-paris-diderot.fr/~henry/ojacare/`

Il nous a été conseillé de nous intéresser à différents documents concernant O'Jacaré [1] et [2], et OCaml-Java [3] et [4].

1.1.2 travail effectué

Mise en place d'un forum de discussion : `http://pstl-interop.forumserv.com`
Mise en place d'un site de gestion de version : `https://github.com/beaCarre/PSTL`

L'installation d'O'Jacaré n'était pas possible, c'est pourquoi nous nous sommes concentrés sur l'étude du projet OCaml-Java.

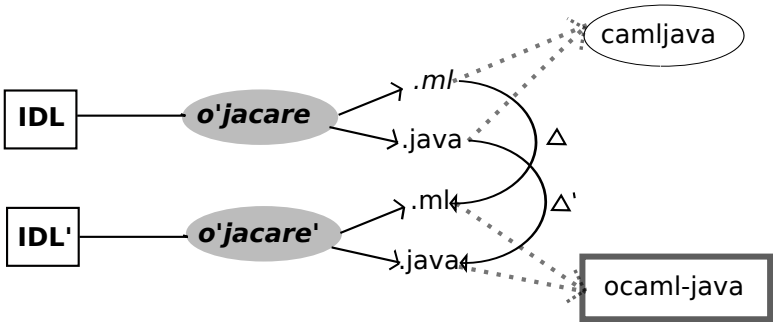
Son code source n'étant pas accessible, nous avons étudié ses outils et manuel utilisateur particulièrement bien détaillés sur le site. Les exemples fournis nous ont beaucoup aidé à comprendre sa méthode, et à maîtriser en partie son utilisation.

Nous avons surtout passé beaucoup de temps à lire des articles (certains en anglais, donc beaucoup plsu de temps). Mais ces articles, très intéressants, nous ont un peu ouvert les yeux sur le sujet du projet et ont répondu à beaucoup de questions restées en attente jusque là sur chacun des outils.

1.2 mercredi 12 février : Découverte d'OCaml Java

1.2.1 sujets abordés

Nous avons fait un point sur la découverte d'Ocaml-Java. La question à se poser est maintenant : comment modifier la génération d'O'Jacaré pour obtenir les classes d'encapsulation adaptées pour OCaml-Java. Dans le Schéma ci-dessus, montrant un schéma



représentant le méchanisme d'O'Jacaré. L'intérêt est de connaître le Δ et Δ'. Plusieurs questions ou sujets ont été lancés :

- encapsulation des classes Java (wrapper) avec attribut callback TODO
- voir héritage multiple O'Caml à partir de classes Java encapsulées TODO
- le polymorphisme dans O'Jacaré TODO
- l'intérêt d'un callback systématique (performances –)
- faire peut-être dans un futur un paquet pour opam ?

Il existe une version d'O'Jacaré ayant été adaptées aux nouvelles version d'OCaml, il nous est désormais accessible. La seconde phase de ce projet sera d'explorer O'Jacaré, son code et ses possibilités.

1.2.2 travail effectué

Nous avons commencé à développer des exemples pour explorer toutes ses possibilités : un othello, en utilisant le OCaml pour le côté calcul, et le Java et sa bibliothèque swing, pour le côté graphique.

L'othello bloqué (pb d'accès à graphics, **A finir si temps**).

O'Jacaré a pu être installé et testé sur les exemples fournis.

Faire othello sur ojacaré TODO

1.3 vendredi 21 février : Découverte d'O'Jacaré

1.3.1 sujets abordés

Question sur l'adaptation pour Ocaml-Java :
faire schemas compilations d'O'Jacaré.
voir pprint, camlp5, yacc et lex,

1.3.2 travail effectué

schemas compil à partir du CIDL, structs de ojacaré (voir annexe)

1.4 lundi 10 mars : Schémas de compilation d'O'Jacaré

1.4.1 sujets abordés

schemas compil KO : départ trop proche code oj, et schémas contiennent trop de code.

=> pas précis et illisible

pb : encapsuler type obj ? => beaucoup de cast ou type plus précis ? => si on arrive à en avoir un self, echappement variable ? alias constructeurs, comment encapsuler ? sous-classe et creer vrai OCaml

1.4.2 travail effectué

schemas compil à partir de la BNF directement, plus formel + ébauche code pour ocaml-java à la main.

1.5 vendredi 21 mars : Schémas de compilation et ébauche adaptation

1.5.1 sujets abordés

schemas compil OK
objet ocaml qui aura uune var d'instance d'objet java ('a java_instance)

la gestion des casts pour accès : faire tous les casts dans les 2 sens ? pour acceder aux super types ?

1.5.2 travail effectué

partir de l'ébauche à la main, et du code généré par o'jacaré, pour rejoindre les deux modèles.

faire exemple sur Point + PC

-> from nothing : tests d'encapsulation de classes simples, pour imaginer la syntaxe : Integer, Float, Number (pour héritgae). (voir t.ml) classe C avec en paramètre p la java$_i$$nstancedelaclasseencapsule, ethritantdelaclasseencapsulantobj, ensous-typantleparamtrepenjava_i$

->from Ojacare : en partant du fichier généré par O'Jacaré : Il est immédiatement ramrquable que un bon nombre d'éléments est inutile. En effet, CamlJava ne fait aucune vérification dynamique de type, ou d'existance d'éléments, que fait OCaml-Java. Ces éléments sont alors inutiles :

```
let clazz =
    try Jni.find_class "mypack/Point"
    with | _ -> failwith "Class␣not␣found␣:␣mypack.Point."
...
try Jni.get_methodID clazz "eq" "(Lmypack/Point;)Z"
      with
      | _ ->
          failwith
            "Unknown␣method␣from␣IDL␣in␣class␣\"mypack.Point\"␣
                :
␣␣␣␣␣␣␣␣␣␣\"boolean␣eq(mypack.Point)\"."
```

puisque OCaml-Java s'occupe des vérifications dynamiques. => réduction immédiate de moitié du nombre de ligne !

References

1. Cours 10 de MPIL : *Interopérabilité OCaml et Java*, E.Chailloux, lien
2. *O'Jacaré, une interface objet entre Objective Caml et Java*, E.Chailloux - G. Henry
3. *OCaml-Java : Typing Java Accesses from OCaml Programs*, X. Clerc, lien
4. *OCaml-Java : from OCaml sources to Java bytecodes*, X. Clerc, lien
5.