Mêler combinateurs et BNF

pour l'analyse syntaxique en OCaml

LAMA

Laboratoire de Mathématiques

UNIVERSITE CHAMBERY ANNECY DE SAVOIE

Rodolphe Lepigre et Christophe Raffalli

JFLA 2015 - 08/01/2015

Pourquoi un nouvel outil ?

Plein d'outils disponibles en `OCaml` :

- `OCamlYacc` (adaptation de `Yacc`)
- `Menhir` (F. Pottier et Y. Régis-Gianas)
- `Camlp4` (D. de Rauglaudre)
- `Dypgen` (E. Onzon)
- `Planck` (J. Furuse)
- `Ostap` (D. Boulytchev)
- ...

On peut les classifier en deux groupes :

- Systèmes « à la *BNF* »
- Combinateurs d'analyseurs syntaxiques

Systèmes « à la *BNF* »

Dans la lignée de `Yacc` (`OCamlYacc`, `Menhir`, `Dypgen`, `Camlp4`)

Principe :

- Spécification du langage avec une syntaxe *BNF*
- Compilation vers un automate à pile (en général)

Avantages :

- Proche de la spécification du langage (génération de doc)
- Très efficace (automate à pile)

Inconvénients :

- Langage *BNF* / *EBNF* peu intégré à `OCaml`
- Utilisation d'un analyseur lexical

Combinateurs d'analyseurs syntaxiques

Dans la lignée de `Parsec` (`Planck`, `Ostap`)

Principe :

- On utilise des analyseurs syntaxiques « atomiques »
- On utilise des structures d'ordre supérieur pour les « combiner »

Avantages :

- Bien adapté à la programmation fonctionnelle
- Facile à réutiliser et à étendre

Inconvénients :

- Pas d'analyseur lexical (en général)
- Récursion gauche interdite

Compiler une syntaxe *EBNF* vers des combinateurs

Les composants :

- `DeCaP` - une bibliothèque de combinateurs efficace
- Un analyseur syntaxique complet pour `OCaml`
- Un mécanisme (fonctoriel) d'extension de syntaxe
- Une extension *EBNF* pour la définition d'analyseurs syntaxiques
- Utilisation d'expressions régulières (`Str`) pour la définition de terminaux

On retient les idées suivantes :

- Combinateurs (`Parsec`, `Ostap`, `Planck`)
- Syntaxe *EBNF* (`OCamlYacc`, `Menhir`, `Camlp4`)
- Système d'extension de syntaxe (`Camlp4`)
- Gestion des caractères non significatifs (`Dypgen`)

Idées originales

Point fixe avec support de la récursion gauche (Christophe Raffalli)

Amélioration de l'efficacité des combinateurs :

- Stream de lignes comme buffer
- Utilisation de continuations délimitées (Koopman et Plasmeijer)
- « Prédicteur » inspectant un caractère (Swierstra)

Gestion des caractères non significatifs :

- Pas d'analyseur lexical
- Fonction de « blank » de type `buffer -> int -> buffer * int`
- On peut construire une telle fonction à partir d'une grammaire

Combinateur « liste »

```
1  let entier =
2    parser
3    | i:''[-]?[0-9]+'' -> int_of_string i
4
5  let liste pelem sep =
6    parser
7    | EMPTY                          -> []
8    | e:pelem es:{STR(sep) pelem}* -> e :: es
9
10 let liste_entiers = liste entier ","
11
12 let blank = Decap.blank_regexp "[ \t]*"
13
14 let parse =
15   Decap.parse_string liste_entiers blank
16
17 let _ =
18   let l = parse "1, -23, 42, 214, -9, 0" in
19   List.iter (fun i -> Printf.printf "%i\n" i) l
```

La traditionelle calculatrice

```
let parser integer =
  | i:''[0-9]+''              -> int_of_string i

let parser atom =
  | i:integer                 -> i
  | "+" - i:integer           -> i
  | "-" - i:integer           -> -i
  | "(" e:{sum | prod}  ")" -> e
and prod =
  | e:atom es:{"*" prod}*     -> List.fold_left ( * ) e es
and sum =
  | e1:prod "+" e2:sum        -> e1 + e2
  | e1:prod "-" e2:sum        -> e1 - e2
  | e:prod                    -> e

let _ =
  let blank = Decap.blank_regexp ''[ \t]*'' in
  try while true do
    Printf.printf ">> %!";
    let r = Decap.parse_string sum blank (input_line stdin) in
    Printf.printf "%i\n%!" r
  done with End_of_file -> ()
```

Travaux futures

Ce qu'il reste à faire :

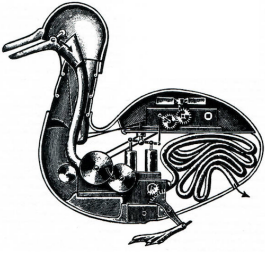
- Support des grammaires récursives à gauche (presque terminé)
- Factorisation à gauche des alternatives
- Finir les « quotations » et les « anti-quotations »
- Terminer la documentation
- Finir le support de `OCaml 4.02` (`#ifversion >= X.XX`)

On a utilisé `DeCaP` pour :

- La grammaire de `OCaml` (`pa_ocaml`)
- La grammaire de `Patoline` (en cours)
- Compiler la base de donnée Unicode en `OCaml`
- Une grammaire `Org-mode` pour `Patoline` (PE Meunier)
- ...

Merci !

HTTP://LAMA.UNIV-SAVOIE.FR/DECAP/



HTTP://WWW.PATOLINE.ORG