Монадические парсер-комбинаторы с поддержкой левой рекурсии

Боровков Д.В., 344 группа
Научный руководитель: к.ф.-м.н., доц. Булычев Д.Ю.

Синтаксический анализ: парсер-комбинаторы

Парсер-комбинаторы — подход к синтаксическому анализу в функциональной парадигме программирования

- Простота
- Композициональность
- Нисходящий анализ => не способны работать с левой рекурсией

```
instance Monad Parser where
   return a = Parser (\cs -> [(a,cs)])
   p >>= f  = Parser (\cs -> concat [parse (f a) cs' |
                                     (a,cs') <- parse p cs])
```

Синтаксический анализ: парсер-комбинаторы

- Комбинаторы alt и seq
- Грамматика N = {S}, T = {‘a’}, R = {S -> S’a’; S -> ’a’}

```
13  let rec S x = alt
14  ................(seq S (terminal 'a'))
15  ................(terminal 'a') x
16  in seq S eof
```

Ostap

- Синтаксическое расширение для работы с парсер-комбинаторами в Objective Caml
- Грамматики с леворекурсивными правилами не поддерживаются

```
let rec lambdaExpr s =
  expr [|left, ["", (fun x y -> `Appl (x, y))]|] p s
and ostap (
  p: x:IDENT                {`Var x}
  |  "\\" x:IDENT "." t:p {`Abstr (x, t)}
  |  -"(" lambdaExpr -")"
)
```

Постановка задачи

- Изучить существующие решения
- Реализовать комбинаторы в рамках библиотеки Ostap
  - Сохранение интерфейсов
  - Поддержка левой рекурсии

Текущая реализация

```
let alt x y s =
  match x s with
  | Failed x ->
      (match y s with
      | Failed y          -> Failed (join x y)
      | Parsed (ok, err) -> Parsed (ok, join x err)
      )
  | x -> x
```

Обзор существующих решений

- “Growing seed”
- Предложено в Packrat Parsers Can Support Left Recursion by Warth 2008
- Недостатки описаны в Direct Left-Recursive Parsing Expression Grammars by Tratt 2010
  - А именно — невозможность поддержать грамматики с правой рекурсией
    рекурсии
  - Tratt предлагает решение части проблем, но класс поддерживаемых грам
    равно меньше КС

Обзор существующих решений

- Подсчет количества заходов в каждый нетерминал
- Описано в Parser Combinators for Ambiguous Left-Recursive Grammars by Frost 2007
- Используется длина входной строки

Обзор существующих решений

- Мемоизация + CPS Style
- Описано в Memoization in Top-Down Parsing by Johnson 1995
- Реализовано и оптимизировано в библиотеке Meerkat (Scala) 2016
- Поддерживается любая КС-грамматика

Реализация

- Для Ostap было решено использовать идею Meerkat
- Используется комбинатор неподвижной точки

```
85  ..let cpsalt a b =
86  ....memo
87  ......(function s ->
88  ........(function k -> begin a s k; b s k end))
89
90  ..let fix2 f =
91  ....let rec p = lazy ((f (function t -> force p t))) in force p
92
93  ..let test4 x = fix2 (function s -> cpsalt (cpsseq s (cpsterminal 'a')) (cpsterminal 'a')) x
94  ..in cpsseq test4 eof (of_string "aaa") (function s -> Printf.printf "Test4::success\n");
```

Реализация

- Таблица для частично примененных парсеров
- Таблица для продолжений
- Таблица для позиций

```
20  ..let memoresult res =
21  ....let rs = ref [] in
22  ....let ks = ref [] in
23  ......function k -> (
24  ........if (List.length !ks = 0)
25  ........then begin
26  ..........ks := k::!ks;
27  ..........let ki = function t -> (if (not (List.mem t !rs))
28  ..................................then begin
29  ....................................rs := t::!rs;
30  ....................................let f elem = elem t in
31  ....................................List.iter f !ks
32  ..................................end) in
33  ..........res () ki
34  ........end
35  ........else begin
36  ..........ks := k::!ks;
37  ..........List.iter k !rs
38  ........end)
39
40  ..let memo f =
41  ....let table = Hashtbl.create 16 in
42  ......function s -> (
43  ........if (Hashtbl.mem table s)
44  ........then begin
45  ..........Hashtbl.find table s
46  ........end
47  ........else begin
48  ..........add table s (memoresult (fun () -> f s));
49  ..........Hashtbl.find table s
50  ........end
51  ........)
```

Результаты

- Изучены существующие решения
- Выбрано наиболее подходящее решение для Ostap
- Реализованы комбинаторы на Objective Caml