Construção de Compiladores I

Capítulo 0

Programando em OCaml

José Romildo Malaquias

Departamento de Computação
Universidade Federal de Ouro Preto

2020.1

1 Introdução

Tópicos

A linguagem OCaml

- **OCaml** é uma linguagem de programação moderna, de alto nível, com muitos recursos úteis.
- Linguagem de uso geral, com ênfase na expressividade e segurança.
- Sistemas de tipos avançado.
- Gerenciamento automático de memória.
- Compilação separada de aplicações autônomas.
- Sistema de módulos sofisticado.
- Estilos de programação funcional, imperativo e orientado a objetos, que facilitam o desenvolvimento de software flexível e confiável.
- `http://ocaml.org/`

Sistema interativo

- OCaml tem um **sistema interativo** chamado de **toplevel** onde o usuário pode digitar frases que são compiladas e executadas.
- As frases do OCaml que o usuário digita devem ser terminadas por `;;` em resposta ao prompt `#`.
- O sistema compila as frases em tempo real, executa-as, e exibe o resultado da avaliação.
- Frases são expressões simples, ou definições `let` de identificadores (valores ou funções).
- O sistema OCaml calcula tanto o valor como o tipo para cada frase.
- Até mesmo os parâmetros de função não necessitam de declaração explícita do tipo: o sistema infere seus tipos a partir de seu uso na função.

Sistema interativo (cont.)

```
# 1+2*3;;
- : int = 7

# let pi = 4.0 *. atan 1.0;;
val pi : float = 3.1415926535897312

# let square x = x *. x;;
val square : float -> float = <fun>

# square (sin pi) +. square (cos pi);;
- : float = 1.
```

Números inteiros e números em ponto flutuante

- **Números inteiros** e **números de ponto flutuante** são tipos distintos, com operadores distintos.
- Por exemplo `+` e `*` operam em números inteiros, mas `+.` e `*.` operam em números em ponto flututante.

```
# 1.0 * 2;;
Error: This expression has type float but an expression was expected of typ
         int
```

Notação de aplicação de função

- Sintaticamente a aplicação de função é indicada escrevendo-se a função seguida dos argumentos.
- Os argumentos não são colocados entre parênteses e nem separados por vírgula.

```
# sqrt 16.0;;
- : float = 4.

# 3. *. sqrt 2.56;;
- : float = 4.8000000000000071

# sqrt 16. +. 1.;;
- : float = 5.

# sqrt (16. +. 1.);;
- : float = 4.1231056256176606
```

Programa

- Códigos em OCaml também podem ser compilados separadamente e executados de forma não interativa utilizando os compiladores de lote `ocamlc` e `ocamlopt`.
- O código-fonte deve ser colocado em um arquivo com extensão `.ml`.
- Ele consiste de uma seqüência de frases, que serão avaliadas em tempo de execução em sua ordem de aparição no arquivo fonte.
- Ao contrário do modo interativo, tipos e valores não são impressos automaticamente; o programa deve chamar as funções de impressão explicitamente para produzir alguma saída.

Exemplo de programa

Exemplo de programa autônomo para imprimir números de Fibonacci:

```
(* Arquivo fib.ml *)

let rec fib n =
  if n < 2 then
    1
  else
    fib (n-1) + fib (n-2)

let main () =
  let arg = int_of_string Sys.argv.(1) in
  print_int (fib arg);
  print_newline ()

let _ = main ()
```

Para compilar e executar o programa:

```
$ ocamlopt -o fib fib.ml
$ ./fib 8
34
```