Prova 1 - Linguagens de Programação

João Antonio Oliveira Pedrosa

Questão 1

1. a) Falso. O uso continua sendo necessário, mesmo para linguagens sem funções de ordem superior.
2. b) Falso. Isso é a definição de sintaxe.
3. c) Falso. Isso é possível em linguagens de escopo dinâmico.
4. d) Falso. Um programa com escopo estático possui uma legibilidade muito maior, pois é mais fácil determinar o estado das variáveis.
5. e) Verdadeiro. A definição de funções de ordem superior é baseada em receber como parâmetro ou retornar funções.
6. f) Verdadeiro. Uma função polimórfica é capaz de operar sobre diferentes tipos de dados.
7. g) Falso. O propósito de semânticas formais é especificar o que as expressões da linguagem significam e como seus significados são compostos através dos significados de suas partes.
8. h) Falso. De acordo com a tése de Church-Turing, cálculo lambda é tão poderoso quanto máquinas de Turing.
9. i) Verdadeiro. Essa é a definição de semântica operacional.
10. j) Falso. Apenas funções que não realizam operações que dependem profundamente do tipo de seus elementos podem ser definidas sem restrições nos tipos dos seus argumentos.

Questão 2

- A) Linguagens tipadas estaticamente são mais seguras contra erros em tempo de execução, pois vários destes são detectados já na compilação. Além disso, o processo de otimização de performance do compilador é facilitado, resultando em uma linguagem mais eficiente.
- B) Com uma semântica formal, podemos argumentar formalmente sobre a linguagem definida.
  Nós podemos, por exemplo, argumentar que dois programas são equivalentes, isto é, provar formalmente que dois programas produzem o mesmo valor se estiverem nas mesmas condições. As provas podem ser feitas por indução nas árvores de derivação que são formadas pelas regras de inferência utilizadas para avaliar a expressão.
- C) Perceba que pela definição do tipo abstrato de dados (uma árvore binária), no final a árvore sempre tem uma folha. Como a função f sempre desce um nível na árvore ao fazer uma chamada recursiva, eventualmente ela irá encontrar uma folha e terminar.
- D) A função f recebe duas listas e retorna a primeira lista invertida e concatenada com a segunda. Por consequência, a função g recebe uma lista e retorna ela com os elementos em ordem inversa.

Questão 3

```
fun f([])   = false |
    f(x::t) = if(x mod 2 = 0) then true else f(t);
```

Questão 4

```
fun concat(list) = foldr (op ^) "" list;

fun getF(l: (string * string) list) =
    (map(fn x => #1 x) l);

fun getS(l: (string * string) list) =
    (map(fn x => #2 x) l);

fun vconc(l: (string * string) list) =
    (concat (getF l), concat (getS l));
```

Questão 5

- O tipo de escopo implementado é o escopo estático, porque o estado das variáveis livres está salvo na Closure da própria função. Assim, a função é executada com as variáveis livres no estado do escopo de onde ela é definida e não sob o estado do escopo de onde ela é chamada.
- Para que o escopo seja dinâmico, a função tem que ser avaliada com as variáveis livres no estado de onde a chamada da função é feita, ou seja, no estado passado por parâmetro para a função junto com a expressão no formato Call(Var f, e).