Tarefa 02 – 2016-02
Tema: Expressões Regulares
Disciplina: Programação Funcional
Professor:Alfio Martini

1 Objetivo

Expressões regulares são padrões utilizados para descrever conjuntos de cadeiaS de caracteres de vários tipos, como:

- identificadores de uma linguagem de programação - cadeias de caracteres alfanuméricos que iniciam com um caracter alfabético.
- constantes numéricas - inteiras ou fracionárias - em uma dada linguagem de programação.
- expressões regulares pode também ser utilizadas para estender a linguagem de padrões em uma determinada linguagem de programação ou editor, permitindo que as funções casem em padrões muito mais complexos do que aqueles já embutidos.

Existem cinco tipos de expressões regulares:

string vazio : ϵ é a expressão regular que casa com o string vazio.

literal : se c é qualquer caracter, c casa com o próprio caracter.

escolha : $r + s$ casa com o padrão definido por r ou s.

concatenação : um string st casa com o padrão rs se st puder ser divido em dois substrings st_1, st_2 tal que $st = st_1 st_2$ e st_1 casa com o padrão de r e st_2 casa com o padrão de t.

iteração : um string st casa com o padrão definido por r^* se ele puder ser subdividido em zero ou mais substrings $st = st_1 \ldots st_n$, tal que cada st_i casa com r. O caso zero implica que o string vazio sempre casa com r^*, para qualquer expressão regular r.

O objetivo desta tarefa é implementar o conceito de expressões regulares em Haskell. *Não utilizar em hipótese alguma funções recursivas na cauda.*

2 Detalhes Específicos

Definir os operadores (construtores) de expressões regulares como funções em Haskell.

A sintaxe das expressões deve ser como mostrado na tabela abaixo:

string vazio	ϵ	eps
literal	c	char
escolha	$r + s$	`r|+|s`
concatenação	rs	`r|.|s`
iteração	r^*	`star r`

Definir também as funções de reconhecimento e de lista de contra-exemplos. A primeira retorna verdadeiro se uma lista de strings está na linguagem descrita pela expressão regular. A segunda recebe uma lista de strings e retorna aqueles que não estão na linguagem definida pela expressão regular. Como exemplos, temos:

```
-- L={aa,ab,ba,bb}
r01 = (a |+| b) |.| (a |+| b)
ss01 = ["aa","ab","ba","bb"]
rec01 = rec r01 ss01 -- retorna True
cex01 = c_exs r01 (ss01 ++ ["aaa","","ab","bba"])
-- retorna ["aaa","","bba"]

-- L = Todos os strings sobre {a,b} que terminam em bb
r03 = star (a |+| b)|.|(b|.|b)
ss03 = ["bb","abb","bbb"]
rec03 = rec r03 ss03 -- retorna True
cex03 = c_exs r03 (ss03 ++ ["aa","ab","ba","bb","aabb"])
-- retorna ["aa","ab","ba"]
```

3 Validação da Implementação

É necessário validar a implementação com expressões regulares para reconhecer as seguintes linguages:

1. Todos strings sobre $\{a, b\}$ com no máximo dois a's.
2. Todos strings sobre $\{a, b\}$ com exatamente dois a's.
3. Todos strings sobre $\{a, b\}$ com nenhum caracter adjacente repetido, isto é, sem nenhum substring da forma aa ou bb.
4. Todos constantes numéricas fracionárias sobre $\{0, \ldots, 9\} \cup \{.\}$.
5. Todos identificadores para tipos em Haskell.
6. Identificadores válidos em Java.

4 Sobre a entrega

- Apresentar um relatório científico descrevendo os detalhes da implementação, em especial da definição dos operadores. Incluir computações detalhadas de alguns exemplos, *em especial dos operador de concatenação e iteração de expressões regulares.*
- O trabalho deve ser feito em grupos de até dois alunos.
- Observar o prazo limite para entrega.
- Para a composição do documento (artigo),**utilizar o sistema de formatação de texto** LaTeX, modelo para artigos da SBC (sociedade brasileira da computação).

- Na sala de entrega, submeter arquivo zipado com relatório científico (em PDF) e código Haskell.
- Todos os integrantes devem submeter o trabalho na sala do Moodle.

5 Material de Apoio

Livros da bibliografia básica e complementar.