1
Introdução

Expressões regulares são um formalismo utilizado para descrever linguagens regulares (HMRU00). Atualmente, diversas linguagens de programação utilizam expressões regulares em suas implementações de casamento de padrão. Porém, expressões regulares puras são muito limitadas para criar alguns padrões práticos e, para contornar essas limitações, muitas implementações de casamento de padrão são baseadas em expressões regulares extendidas – chamados *regexes* – que consistem na combinação de expressões regulares com construções que focam em problemas específicos de casamento de padrão.

No final dos anos 90, os regexes de Perl (WS00) foram amplamente utilizados devido ao poder expressivo resultante do seu conjunto extenso de contruções (Fri06). Posteriormente, outras linguagens de script adotaram os regexes de Perl como base de suas implementações de casamento de padrão, por exemplo, Python (Lut06) e Ruby (TFH09), que possuem implementações compostas por um subconjunto dos regexes de Perl com diferenças sutis na semântica de algumas construções. Em Lua (Ier06), a biblioteca de casamento de padrão é baseada em expressões regulares simplificadas, de modo que só são permitidas alternativas e repetições de classes de caracteres.

Embora auxiliem a construção de padrões expressivos, os regexes têm implementações complexas e distantes do formalismo original de expressões regulares. Devido à ausência de uma base formal, é impossível prever a complexidade de determinados padrões, além de ser difícil inferir a semântica de algumas combinações entre as construções.

Ford (For04) apresentou um formalismo para reconhecimento de linguagens chamado Parsing Expression Grammars (PEG). Uma PEG possui uma sintaxe semelhante à de uma CFG na notação BNF com operações de expressões regulares; porém, ao invés de definir uma linguagem como as CFGs, uma PEG define um reconhecedor *top-down* para essa linguagem (For04).

A motivação para o uso de PEGs como alternativa para gramáticas livres de contexto se deve ao seu poder de expressão. Por exemplo, PEGs podem reconhecer todas as linguagens livres de contexto *LR(k)* determinísticas e algumas linguagens que não são livres de contexto (For04). Além disso,

PUC-Rio - Certificação Digital Nº 0812596/CA

PEGs possuem um modelo formal que descreve claramente a semântica de qualquer gramática. Recentemente, PEGs foram propostas como alternativa para casamento de padrões (Ier09), já que PEGs são mais expressivas que expressões regulares e possuem operadores similares aos de expressões regulares e regexes, como lookaheads e quantificadores possessivos.

A ausência de formalismo dos regexes e a expressividade de PEGs nos motivou a investigar correspondências entre regexes e PEGs. Um dos benefícios de converter regexes em PEGs é o melhor entendimento da semântica dos regexes, baseado no modelo formal de PEGs. Outro resultado interessante é a possibilidade de executar regexes sobre implementações de PEGs e oferecer uma base teórica para possíveis otimizações.

O objetivo deste trabalho é estudar a conversão de regexes para PEGs. Um dos resultados deste trabalho foi a criação de uma função baseada em continuação que converte expressões regulares puras em PEGs (OIdM10).

Para convertermos regexes em PEGs é necessário conhecer a semântica de cada construção regex. Para isso, apresentamos um estudo das construções regexes de Perl (WS00), PCRE (Haz09) (Perl Compatible Regular Expressions), Python (Fou10), Ruby (TFH09) e Lua (IdFC06), analisando a semântica das principais construções, como capturas, lookahead, repetição possessiva e expressões independentes. Este estudo também lista as construções que não conseguimos converter para PEGs, por exemplo, lookbehinds e backreferences, e aponta as características destas construções que dificultam a sua conversão para PEGs.

Esse trabalho também apresenta a implementação de um conversor de regexes para PEGs chamado *Lua Regex*. Para converter regexes em PEGs, o Lua Regex implementa a função baseada em continuação e suas extensões apresentadas neste trabalho. Esse conversor aceita como entrada expressões compostas por um subconjunto dos regexes de Perl e os regexes de Lua. A conversão resulta em uma gramática de LPeg (Ier09), que é uma implementação de PEGs para a linguagem Lua. A idéia básica desse conversor é permitir que desenvolvedores continuem utilizando a sintaxe de regexes, que é amplamente conhecida, deixando transparente que a sua execução seja sobre uma implementação de PEGs, no caso LPeg.

Este trabalho está organizado da seguinte forma: no capítulo 2 apresentamos um estudo detalhado dos regexes de Perl, PCRE, Python, Ruby e Lua baseado nas suas respectivas documentações. Para enriquecer esse estudo, fizemos diversos testes que visavam comprovar (ou confrontar) o que foi definido em cada documentação. O resultado desse estudo nos permite definir quais são as construções mais interessantes para a conversão e quais não serão

PUC-Rio - Certificação Digital Nº 0812596/CA

abordadas por serem pouco utilizadas na prática. No capítulo 3 descrevemos o formalismo de PEGs, suas vantagens em relação a expressões regulares e uma breve introdução a uma implementação de PEGs para a linguagem Lua, chamada LPeg (Ier08). No capítulo 4 introduzimos uma função baseada em continuação que converte expressões regulares em PEGs, e extensões dessa função que convertem um subconjunto de regexes. No capítulo 5 apresentamos o *Lua Regex*, uma implementação prática do continuation-based conversion para a linguagem Lua. Finalmente, no capítulo 6 discutimos as conclusões deste trabalho e resumimos suas contribuições.

PUC-Rio - Certificação Digital Nº 0812596/CA