UniRuy & Área 1 | Wyden

PROGRAMA DE ENGENHARIA DA COMPUTAÇÃO

TEORIA DE COMPILADORES

JOÃO VICTOR DE DEUS MARTINS

Teoria de Compiladores: LINGUAGENS, EXPRESSÕES E GRAMÁTICAS REGULARES

Salvador - Bahia - Brasil

2022

JOÃO VICTOR DE DEUS MARTINS

Teoria de Compiladores: LINGUAGENS, EXPRESSÕES E GRAMÁTICAS REGULARES

Trabalho Acadêmico elaborado junto ao programa de Engenharia UniRuy & Área 1 | Wyden, como requisito para obtenção de nota parcial da AV1 na disciplina Teoria de Compiladores no curso de Graduação em Engenharia da Computação, que tem como objetivo consolidar os tópicos do plano de ensino da disciplina.

Orientador: Prof. MSc. Heleno Cardoso

Salvador - Bahia - Brasil
2022

TERMO DE APROVAÇÃO

JOÃO VICTOR DE DEUS MARTINS

TEORIA DE COMPILADORES: LINGUAGENS, EXPRESSÕES E GRAMÁTICAS REGULARES

Trabalho Acadêmico aprovado como requisito para obtenção de nota parcial da AV1 na disciplina Teoria de Compiladores, UniRuy & Área 1 | Wyden, pela seguinte banca examinadora:

BANCA EXAMINADORA

Profº. MScº. Heleno Cardoso
Wyden

Salvador, 05 de Novembro de 2022

Dedico este trabalho acadêmico a todos que contribuíram direta ou indiretamente com minha formação acadêmica.

Agradecimentos

Primeiramente agradeço a Deus. Ele, sabe de todas as coisas, e através da sua infinita misericórdia, se fez presente em todos os momentos dessa trajetória, concedendo-me forças e saúde para continuar perseverante na minha caminhada.

E a todos aqueles que contribuíram direta ou indiretamente para a minha formação acadêmica.

"A educação tem raízes amargas, mas os seus frutos são doces".

Aristóteles.

Resumo

Funções de ordem superior são expressões que podem conter outras expressões como um elemento central. Uma vez construídas, essas expressões podem conter outras expressões de forma auto-recursiva. Nós nos referimos a elas como expressões regulares recursivas. Podemos construir autômatos auto-replicantes a partir deste material. Essas construções exemplificam o uso de expressões regulares de ordem superior, bem como a capacidade de aceitar gramáticas livres de contexto sem limites para seu estado. Autômatos com propriedades recursivas podem ser construídos a partir deste material.

Palavras-chaves: Linguagens regulares, Expressão regular, Operações regulares, Gramática regular, autômato finito.

Abstract

Higher-order functions are expressions that can contain other expressions as a central element. Once constructed, these expressions can contain other expressions in an auto-recursive manner. We refer to them as recursive regular expressions. We can build self-replicating automata out of this material. These constructs exemplify the use of higher-order regular expressions, as well as the ability to accept context-free grammars without limits to their state. Automata with recursive properties can be built from this material.

Keywords: Regular languages, Regular expression, Regular operations, Regular grammar, finite automaton.

Lista de abreviaturas e siglas

DFAs	- Deterministic Finite Automaton (Autômato Finito Determinístico).
Grammars	- Gramática.
NFAs	- Nondeteterministic Finite Automata (Autômatos finitos não determinísticos).
String	- Cadeias de caracteres que armazenam dados textuais.

Sumário

[1] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

1 LINGUAGENS, EXPRESSÕES E GRAMÁTICAS REGULARES

1.1 Introdução

Dados de string são qualquer sequência de caracteres ou bytes com os quais um programa pode trabalhar. Os programas podem entender strings por meio do uso da gramática. A gramática ajuda os programas a distinguir entre sequências de strings legais e ilegais. Quando usada, a gramática divide strings em estruturas de dados com as quais os programas podem trabalhar. As gramáticas criam estruturas de dados que geralmente são recursivas. Essa é uma característica comum dos dados gerados pela gramática e é discutida na revisão. Outro tipo de gramática é chamado de expressão regular; ele foi projetado especificamente para uso no processamento de texto. As expressões regulares são populares para executar várias outras tarefas de manipulação de strings. Isso inclui extrair informações do texto, reorganizar suas palavras ou caracteres e muito mais.

1.2 Execução/Método

Resenha desenvolvida através de analises de artigos científicos apurados.

1.2.1 Repositório de Pesquisa

IEEE

1.2.2 String de Busca por Repositório

"Expressions and Regular Grammars"
"Languages, Expressions and Regular Grammars"

1.2.3 Artigos Selecionados

Higher order regular expressions
Analysis of grammatical evolutionary approaches to regular expression induction
Creating the English Grammar Tenses Pattern Using Regular Expression Method

1.2.4 Resenha dos Artigos Selecionados

Expressões regulares são usadas por programadores para criar padrões de pesquisa muito específicos. Eles também são amplamente usados em programação para definir linguagens. Somando-se aos métodos já conhecidos de criação de linguagens estão DFAs, Grammars e NFAs. Descrições em texto simples em inglês ou qualquer outro idioma comum são difíceis de implementar e imprecisas. Usar um DFA ou NFA geralmente requer um editor de texto gráfico que leva tempo para aprender. As expressões regulares são fáceis de digitar, o que as torna mais fáceis de implementar do que descrições longas. Eles tendem a ser gerais o suficiente para representar idiomas populares sem exigir muito tempo para aprender. As expressões regulares facilitam o pressionamento das teclas quando você tem uma palavra ou frase específica em mente. Mesmo com uma palavra difícil de expressar, as expressões regulares podem ser digitadas rapidamente. As expressões regulares são úteis para pesquisar texto em muitos aplicativos. Eles também são usados para pesquisar arquivos na linha de comando e em editores de texto para textos ou strings específicos. Leva anos de estudo para aprender um novo idioma. Expressões regulares parecem aprender uma linguagem diferente porque leva anos para dominar. Depois que alguém aprende expressões regulares, pode reduzir o tempo necessário para trabalhar no texto em milhares de horas. Este exemplo mostra uma expressão regular com componentes rotulados. fechar em ação definida Uma linguagem L é uma linguagem regular se existe um autômato finito M que determina a linguagem. Como a linguagem L é um conjunto de strings, uma maneira de criar uma nova linguagem é usar operações de conjunto padrão, como união e interseção, para criar um novo conjunto. Uma questão importante nessas operações é se o novo idioma formado pela operação é regular se os dois idiomas originais forem regulares. Na palestra de hoje, veremos que para cada operação padrão, incluindo união, complemento e interseção, se a linguagem original envolvida na operação for regular, a nova linguagem formada por essa operação será regular.

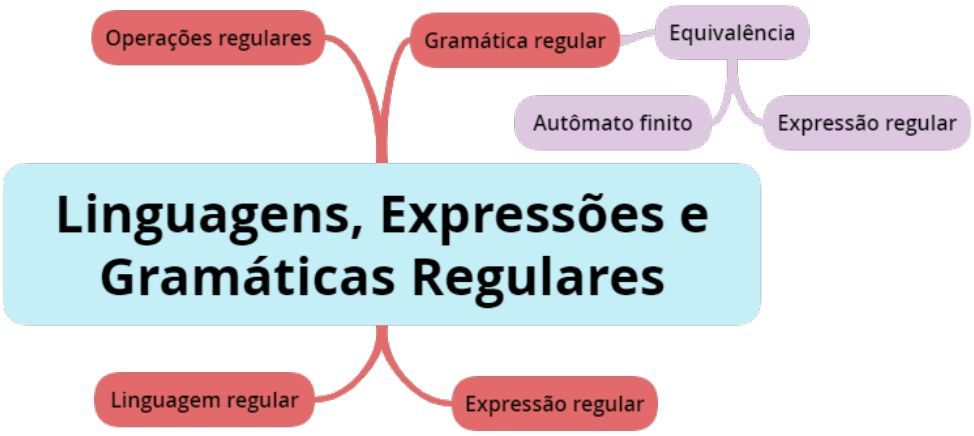
Teorema Se L1 e L2 são linguagens regulares, então a nova linguagem L = L1 L2 é uma linguagem regular. Prova Como L1 é regular, existe um DFA M1 que determina o idioma. Como L2 é regular, existe um DFA M2 que determina o idioma. Esta é a estrutura de uma máquina M que pode decidir L = L1 L2 Complemento Teorema Se L é uma linguagem regular, seu complemento L também é regular. Prove que se L é regular, existe um DFA M que determina L. Construa um novo estado de máquina M de M alterando

todos os estados de não aceitação em M para estados de aceitação e alterando todos os estados de aceitação em M para estados de não aceitação. Para qualquer sequência de entrada x L, x leva M para o estado de aceitação. O mesmo x colocará M em um estado de não aceitação. Da mesma forma, uma string de entrada x que não seja L levará M a um estado de não aceitação, o que significa que o mesmo x leva M a um estado de aceitação. Portanto, M só aceitará x se x L.

Se L é uma linguagem, então a linguagem L* são todas as linguagens que podem ser escritas como uma concatenação de 0 ou mais strings extraídas de L. Teorema Se L é uma linguagem regular, então L* também é uma linguagem regular. Prove que se L é regular, existe um DFA M que determina L. Expressões comuns Expressões regulares são expressões formadas pegando strings e combinando-as com um conjunto de operadores: Em série: R = R1R2 União: R = R1 R2 Intersecção: R = R1 R2 Kleene: R=R1* Complemento de dois: R = R1 Por exemplo, considere R = 01 00 (10)*11). Teorema Uma linguagem é regular se e somente se seus elementos podem ser descritos por expressões regulares. Prova Seja L o conjunto de strings gerado por alguma expressão regular R. Os elementos básicos em expressões regulares são todos strings simples: um conjunto consistindo apenas dessas strings é um idioma regular, porque é fácil construir um DFA que aceita apenas strings e nada. Como todas as operações regulares usadas para combinar essas strings em expressões regulares correspondem diretamente a operações regulares em conjuntos, a combinação das strings base produz automaticamente uma linguagem regular. As gramáticas formais definem formalmente a estrutura de uma linguagem definindo suas strings. Eles normalmente se concentram em gramáticas livres de contexto, também conhecidas como spanners. No entanto, as expressões regulares compartilham o mesmo núcleo gramatical formal. Nesse caso, as expressões regulares são definidas por gramáticas formais que também são chamadas de gramáticas regulares. Um formalismo gramatical G deve incluir um conjunto de quatro elementos. G é uma abreviatura de (N, T, P, S), que se refere a um conjunto. Uma lista chamada S inicia o processo. Esta lista contém o símbolo inicial, que é chamado de N. N é um conjunto de símbolos não terminais e T é um conjunto de símbolos terminais. Ao lado de N está P, que contém regras de produção. entender a gramática de uma linguagem é entender uma estrutura que pode ser referenciada várias vezes. Isso permite que programas como analisadores ou regexes reconheçam ou filtrem strings com base na gramática. Diferentes estilos gramaticais usam metodologias diferentes. Por exemplo, gramáticas regulares usam autômatos finitos como mecanismo

de implementação. Por outro lado, gramáticas sem contexto usam “sparers” para analisar mensagens; esse estilo é conhecido como autômatos push-down. Os computadores usam padrões e instintos de geração em quase todos os aspectos de sua operação. Os programas incluem símbolos identificáveis como parte de seu design. as regularidades nas palavras de um texto. o mesmo caractere aparece com frequência nos dados de um arquivo. o arranjo de unidades menores em uma imagem. Máquinas formais (ou abstratas) são usadas por editores de texto e compiladores para reconhecer padrões em suas entradas. Essas máquinas reconhecem padrões automaticamente sem nenhuma instrução específica. A escrita formalizada refere-se a linguagens de programação e comandos que usam expressões regulares. Eles são usados em manuais de linguagem de programação para explicar os padrões legais de entrada de dados. Eles também são usados no shell do Unix para definir padrões para nomes de arquivos, etc. Assim como os autômatos finitos são usados para identificar padrões de strings, expressões regulares são usadas para gerar padrões de strings. Uma expressão regular é uma fórmula algébrica cujo valor é um padrão que consiste em um conjunto de strings, chamado de linguagem de expressão. Os operadores em expressões regulares podem ser: Caracteres do alfabeto que definem a expressão regular. Uma variável cujo valor é qualquer padrão definido pela expressão regular. epsilon representa uma string vazia sem caracteres. null significa um conjunto de strings vazio. Os operadores usados em expressões regulares incluem: Um símbolo de tubo duplo literal pode ser representado por R1 e R2, ou R1 U R2. Ambas as expressões são expressões regulares e, quando usadas juntas, são ainda mais poderosas. L(R1 | R2) = L(R1) + L(R2). R1 e R2 são expressões regulares; então R1.R2 é uma expressão regular. Alternativamente, R1R2 é escrito como R1.R2. O L medido de R1 e R2 combinados para formar um único comprimento. Um encerramento de Kleene é uma expressão regular que é o resultado da interpretação de uma expressão regular R1 . Qualquer expressão regular R1 * R1 também é uma expressão regular. L(R1) = épsilon U L(R1R1) U L(R1R1R1) U ... L(R1*) L(R1) = épsilon U L(R1R1) U L(R1R1R1) U ... A conversão para um autômato finito pode facilitar a aceitação de uma linguagem. Alguns algoritmos de conversão são mais complicados, mas versões simples estão disponíveis. O algoritmo de construção 5 para converter autômatos finitos em gramáticas lineares à esquerda é explicado na íntegra. Além disso, sua prova de correção é apresentada e um exemplo relevante é fornecido.

Mapa Mental



1.2.5 Perguntas e Respostas

1. para quê são usados os autômatos finitos?

Identificar padrões de strings.

2. O que é uma expressão regular?

Uma expressão regular é uma fórmula algébrica cujo valor é um padrão que consiste em um conjunto de strings, chamado de linguagem de expressão.

3. Para que serve uma Expressões regular?

regulares são usadas por programadores para criar padrões de pesquisa muito específicos. Eles também são amplamente usados em programação para definir linguagens.

1.3 Conclusão

Como os autômatos finitos são usados para reconhecimento de padrões, expressões regulares são usadas para geração de padrões. "Assim como os autômatos finitos são usados para identificar padrões de strings, expressões regulares são usadas para gerar padrões de strings. Expressões regulares produzem padrões. Dada a contribuição, este modelo gera um processo de tomada de decisão. Este processo de decisão produz um resultado booleano.

Referências[1]

SIČÁK, M. Higher order regular expressions. jul. 16DC.

GONZALEZ PARDO, A.; CAMACHO, D. Analysis of grammatical evolutionary approaches to regular expression induction. jul. 14DC.

BROER BAHAWERES, R.; PRASETYO, B.; BAYU SUSENO, H. Creating the English Grammar Tenses Pattern Using Regular Expression Method. jan. 23DC.

[1] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.