VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Vizualizace regulárních výrazů

Regular Expression Visualization

Dominik Kundra

Bakalářská práce

Vedoucí práce: Ing. Jakub Beránek

Ostrava, 2024

VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Č.j.VSB/23/113247
vsboee23037B25

Zadání bakalářské práce

Student: **Dominik Kundra**

Studijní program: B0613A140014 Informatika

Téma: Vizualizace regulárních výrazů
Regular Expression Visualization

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit nástroj sloužící pro vizualizaci a ladění regulárních výrazů. Nástroj by měl být schopný zpracovat zvolený regulární výraz, sestavit plán vykonávání daného výrazu dle zvolené implementace a poté umožnit programátorovi interaktivně krokovat provádění regulárního výrazu. Nástroj by měl být vytvořen jako rozšíření do vývojového prostředí (např. do Visual Studio Code), aby šel jednoduše použít při vývoji programů. Výsledná aplikace by měla být řádně zdokumentována a při jejím vývoji by měl být využit verzovací systém (např. git).

1. Analyzujte a popište možnosti implementace regulárních výrazů.
2. Navrhněte architekturu rozšíření do vývojového prostředí, které bude schopné analyzovat regulární výrazy ze zvoleného zdrojového kódu.
3. Naimplementujte nástroj pro vizualizaci regulárního výrazu a integrujte jej do vývojového prostředí.
4. Otestujte vizualizaci nástroje na regulárních výrazech z reálných projektů.

Seznam doporučené odborné literatury:

[1] FRIELD, Jeffrey. Mastering Regular Expressions 3rd Edition. 2006. O'Reilly Media. ISBN: 978-0596528126
[2] SORVA, Juha. Visual program simulation in introductory programming education. Espoo: Aalto Univ. School of Science, 2012. ISBN 9789526046266. Dostupný také z WWW: http://doi.acm.org/10.1145/2445196.2445368.
[3] VANDERKAM, Dan. Effective TypeScript : 62 Specific Ways to Improve Your TypeScript. O'Reilly Media, 2019. ISBN 978-1492053699

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jakub Beránek**

Datum zadání: 01.09.2023

Datum odevzdání: 30.04.2024

Garant studijního programu: doc. Mgr. Miloš Kudělka, Ph.D.

V IS EDISON zadáno: 09.11.2023 15:43:31

Abstrakt

No Czech or Slovak abstract is given

Klíčová slova

No Czech or Slovak keywords are given

Abstract

No English abstract is given

Keywords

No English keywords are given

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Vyhledávání v textu patří mezi základní problémy, se kterými se velice pravděpodobně potká skoro každý programátor. Tento problém se dá řešit mnoha způsoby, avšak ne všechna řešení lze použít univerzálně a každý ze způsobů má své výhody a nevýhody. Jedním ze přístupů je využití regulárních výrazů. Jedná se o sadu znaků, které nám umožňují nadefinovat výraz a ten je následně převedený na strukturu, nejčasteji ve formě konečných automatů. Téměř každý dnešní programovací jazyk je obsahuje, ale jejich implementace se mohou lišit.

Cílem této práce je naimplementovat nástroj, který bude umožňovat procházení regulárních výrazů a následné vizualizování těchto průchodů, jako rozšíření do vývojového prostředí.

Při vývoji programů, je programátor často obeznámen s regulárními výrazy, jedná se totiž o poměrně rychlé řešení pro vyhledávání v textu. Můžeme se s nimi setkat v podstatě skoro ve všech částech softwaru[1], např. validace formulářů, vyhledávání v textu nebo třeba v příkazovém řádku. Tyto výrazy se však brzy mohou stát hůře čitelnými, jelikož neumožňují žádné formátování[2]. Taktéž mohou být pro mnoho lidí matoucí, či nepřehledné. Z tohoto důvodu se hodí mít nástroj, který potencionálně usnadní práci programátorům, aby si mohli zobrazit průchod zadaným výrazem. Dále pro lidi, kteří například vidí tyto výrazy poprvé v životě může být snažší jim porozumět, je-li jim ukázáno jak fungují ve krocích. Sice již existují řešení tohoto problému a to v různých formách [1, 2, 3], ale pro zvolené vývojové prostředí mnoho přístupů neexistuje. Tato situace je motivací, zabývat se problémem do hloubky a nabídnout originální řešení v daném směru, které by mohlo být přínosem pro programátory.

[1] počítačový program, aplikace
[2] upravení vzhledu, tvaru

Kapitola 2

Princip fungování regulárních výrazů

Abychom mohli porozumět, jakým problémem se v této práci budeme zabývat, musíme si zadefinovat co jsou regulární výrazy, jak fungují a jak se jednotlivé implementace mohu lišit. Následovně si vysvětlíme, jak lze zjednodušit pochopení těchto výrazů, popřípadě jak rychle najít chybu ve vlastním výrazu.

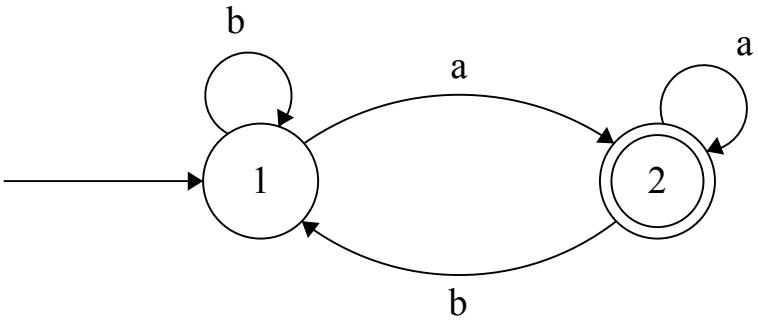
2.1 Definice

Předtím než si vysvětlíme blíže jak fungují samotné regulární výrazy, je potřeba si první objasnit význam několika pojmů z teoretické informatiky. Regulární jazyky jsou jednou z možných definic formálních jazyků.

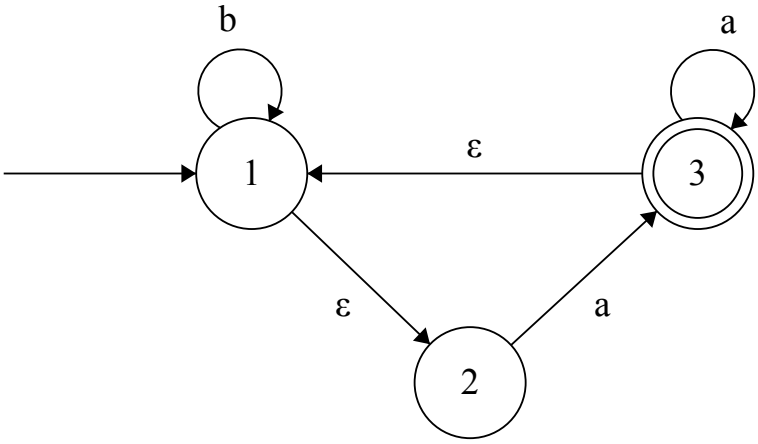
Formální jazyk je libovolná množina koečných slov nad určitou abecedou. Slova chápeme jako retězece znaků, která jsou příjimaná zadaným jazykem. Tato slova musí být sice konečná ale množina těchto slov může být nekonečná. Tyto jazyky mohou být definovány regulárními výrazy, formální gramatikou, konečnými automaty a dalšími.

Ve spojení regulárních výrazu se často vyskytují konečné automaty, jedná se o další oblast v teoretické informatice. Proto abychom pochopili proč je tato oblast pro nás důležitá, musíme si vysvětlit co vlastně jsou.

Konečné automaty popisují jsou modely jednoduchého počítače, který má určitý počet stavů a přechodů [4]. Dělíme je na deterministické a nedeterministické, zkráceně DFA (deterministic finite automata) a NDA (non-deterministic finite automata). Deterministické automaty mohou mít v daném stavů pro každý znak abecedy maximálně jeden přechod, dále nemohou obsahovat tzv. prázný znak často označovaný řeckým písmenem epsilon ϵ. Nedeterministické naopak obojí umožňují, prázné znaky nám umožňují změnu stavu bez změny aktuální pozice v hledaném slově. Každý NFA lze převést na ekvivalentní DFA.



Obrázek 2.1: Příklad deterministického automatu příjamjící slova obsahující písmena z abecedy {a, b} končící písmenem a



Obrázek 2.2: Příklad nedeterministického automatu ekvivalentního k předchozímu deterministickému

2.2 Původ a vznik

Regulární výrazy byly poprvé nadefinovány Americkým matematikem **Stephan Cole Kleene**, jako regulární jazyky. Dále se aplikovali v teoretické informatice, jako podkategorie **teorie automatů** a součást **formálních jazyků**. Ačkoliv byli nadefinovány začátkem padesátých let, tak jejich využití v počítačích nastalo až na konci šedesátých let a to v jedním z nejznámějších operačních systémů UNIX.

Ken Thompson byl tím kdo navrhnul první implementaci využívanou v počítačích, která se používá do dnes. Tento algoritmus se pojmenoval **Thompson's construction** (Thompsonovo sestavení), který převádí textovou reprezentaci výrazu na ekvivalentní nedeterministický automat.

Literatura

1. DIB, Firas. *Build, test, and debug regex* [online]. [B.r.]. [cit. 2024-01-25]. Dostupné z: `https://regex101.com/`.
2. AVALLONE, Jeff [online]. [B.r.]. [cit. 2024-01-25]. Dostupné z: `https://regexper.com/`.
3. [online]. [B.r.]. [cit. 2024-01-25]. Dostupné z: `https://regexr.com/`.
4. HAVRLANT, Lukáš. *Konečný Automat* [online]. [B.r.]. [cit. 2024-02-06]. Dostupné z: `https://www.matweb.cz/konecny-automat/`.