A black background with a black square

Description automatically generated with medium confidence **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**

ФАКУЛТЕТ ПО КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ

КОМПЮТЪРНО И СОФТУЕРНО ИНЖЕНЕРСТВО

**Формални езици и езикови процеси:**

Курсова работа:

„ Графично представяне на регулярни изрази. Синтактичен граф генератор“

|  |  |  |
| --- | --- | --- |
| Съставил: | Факултетен номер: | Група: |
| Ангел Любомиров Стойнов | 121222150 | 40 |

Съдържание

[Въведение: 3](#_Toc195126988)

[История: 4](#_Toc195126989)

[Защо го ползваме: 5](#_Toc195126990)

[Сравнение: 6](#_Toc195126991)

[Синтактичен граф генератор: 6](#_Toc195126992)

[Генератор на railroad диаграми: 6](#_Toc195126993)

[Заключение: 7](#_Toc195126994)

[Документация по кода: 7](#_Toc195126995)

[Реализация: 7](#_Toc195126996)

[Използвана литература: 7](#_Toc195126997)

# Въведение:

Регулярните изрази са мощен инструмент за описание и разпознаване на текстови модели. Те представляват последователности от символи, които дефинират шаблони за търсене, валидиране или манипулиране на текстови данни. Чрез регулярни изрази можем да проверяваме дали даден низ отговаря на определен формат, да извличаме специфични части от текст или да заменяме сегменти от него.​

**Основни операции при регулярните изрази:**

* **Дизюнкция (Alternation):** Представлява избора между два или повече варианта. Например, изразът a|b ще съвпадне с 'a' или 'b'.​
* **Конкатенация (Concatenation):** Последователно свързване на символи или изрази. Например, ab ще съвпадне с низа 'ab'.​
* **Итерация (Kleene Star):** Позволява повторение на предходния елемент нула или повече пъти. Например, a\* ще съвпадне с '', 'a', 'aa', 'aaa' и т.н.​

Регулярните изрази намират широко приложение в различни области на програмирането и текстообработката. Те се използват в текстови редактори за функции като "търсене и замяна", в програми за валидиране на входни данни, както и в инструменти за анализ и обработка на текст. Много езици за програмиране, включително Python, JavaScript и C#, предоставят вградена поддръжка за работа с регулярни изрази, което улеснява разработчиците при изпълнението на сложни текстови операции.

Синтактичните диаграми са начин за представяне на безконтекстна граматика. Те представляват графична алтернатива на формата на Бекус-Наур (BNF), разширения BNF (EBNF), допълнения BNF (ABNF) и други текстови граматики, използвани като метаезици.

Ранни книги, използващи синтактични диаграми, включват „Ръководство за потребителя на Pascal“, написано от Никлаус Вирт, както и ръководството Burroughs CANDE Manual.

В областта на компилацията: текстовите представяния, като BNF и неговите варианти, обикновено се предпочитат. BNF е базиран на текст и се използва от разработчиците на компилатори и генератори на парсъри.

Диаграми са визуални и могат да бъдат по-лесно разбрани от неспециалисти, поради което понякога се използват и в графичен дизайн. Официалният документ, дефиниращ формата за обмен на данни JSON, е пример за съвременно и широко използване на такива диаграми.

# История:

Стивън Клийн изобрети регулярните изрази в средата на 50-те години на миналия век, като обозначение за крайни автомати. Всъщност, те са еквивалентни на крайните автомати по отношение на това, което представят – формално описание на множеството от низове, които автоматът може да разпознае.

Първоначално регулярните изрази се появяват в програмен контекст във версията на текстовия редактор QED, разработена от Кен Томпсън, през средата на 60-те години. През 1967 г. Томпсън подаде заявка за патент за механизъм за бързо съвпадение на текст, базиран на регулярни изрази. Патентът беше издаден през 1971 г. и се счита за един от първите патенти за софтуер.

Тези иновативни идеи не само положиха основите за теорията на формалните езици и крайните автомати, но и значително повлияха на развитието на инструменти за текстова обработка. Работата на Томпсън допринесе за създаването на Unix-базирани програми, като grep, sed и awk, които и до днес са ключови в анализа и манипулирането на текстови данни.

# Защо го ползваме:

Регулярните изрази и синтактичните диаграми намират широко приложение поради следните причини:

* **Ефективност и прецизност:** Регулярните изрази позволяват бързо и точно търсене и обработка на текстови данни, като разпознават специфични шаблони без необходимост от сложна логика.
* **Гъвкавост в обработката на данни:** С тях може лесно да се проверява валидността на входни данни (например имейли, URL адреси, телефонни номера), да се извличат конкретни части от текст или да се извършват замени.
* **Приложение в множество технологии:** Много езици за програмиране – като Python, JavaScript, C# – вграждат поддръжка за регулярни изрази, което улеснява бързото им интегриране във всякакви приложения.
* **Подобряване на визуализацията на граматики:** Синтактичните диаграми предоставят визуално представяне на граматиката, което може да бъде по-разбираемо за неспециалисти и полезно при обучението или дизайна на езици за програмиране.
* **Поддръжка при разработката на инструменти:** Те се използват от генератори на парсъри и компилатори за визуализиране на синтактичната структура на езиците, което спомага за по-доброто разбиране и поддръжка на сложни системи.

# Сравнение:

## Синтактичен граф генератор:

* Този инструмент взема граматика (например дефинирана чрез регулярни изрази или контекстно-свободна граматика) и я представя във вид на граф (или дърво), където възлите представляват граматически конструкции (като нетерминали и терминали), а ребрата – връзките между тях.
* Цел: Да даде структурно и детайлно визуално представяне на синтаксиса, което може да помогне за анализиране и отстраняване на грешки в граматиката.

## Генератор на railroad диаграми:

* Този инструмент създава диаграми, които имитират железопътни схеми, където посоките и клонищата показват потока на граматическите правила. Railroad диаграммите са създадени така, че да бъдат по-интуитивни и лесни за възприемане, особено от неспециалисти.
* Цел: Да направи граматиката по-разбираема чрез визуална лекота, като представи опциите (напр. алтернативи и последователности) по начин, който е удобен за потребителите, които не са дълбоко запознати с формалната теория.

# Заключение:

# Документация по кода:

# Реализация:

# Използвана литература: