Крайни автомати

Проект на Йоан Георгиев Саламбашев, КН 2, ф.н. 82063, 2020/2021г.

1. **Увод**

Проектът има за цел да представи детерминирани и недетерминирани крайни автомати. За тях са реализирани основните операции, от които те се нуждаят, както и други помощни методи. Разнообразието от методи позволява на клиента да използва пълноценно без каквито и да е притеснения функционалността на крайните автомати. Библиотеката е лесна за употреба и предразполага използващия ги към приятна работа с всяка една основна операция. Създаването на такава библиотека за управление води до възможността да се правят различни реализации и визуализации на крайни автомати. Капсулацията на данните гарантира сигурност и липсата на възможност за злоупотреба.

Командите, които библиотеката предоставя, са предназначени да улеснят максимално клиента и да не му се налага да мисли дали има нещо неправилно или невалидно сред данните за автоматите. Това се осигурява, заради последователното изпълнение на всяка команда и липсата на голямо натоварване на устройството, на което се изпълнява. Това обаче никак не означава, че обхватът на командите е малък. Една команда може както да промени минимално количество от данни, така може и да промени цялата информация за един автомат. Всичко е синхронизирано с нуждите на клиента. Макар и да е възможно потребителят да допусне грешка и да обработи нежелената от него информация, той може да не запази последните промени и да възстанови миналото състояние на данните само с последователно изпълнение на 2 команди.

1. **Проектиране и структура**
2. За реализацията на проекта е използван голям обхват от структури от данни на езика С++. Четим код и спестяване на памет и ресурси се осигурява чрез разделянето на кода в множество функции и имплементирането на оптимални конструкции за необходимите цели. Разнообразието им позволява всичко необходимо за ефективна работа с 3 вида автомати. Всички те са реализирани в различни класове и всеки спазва основните ООП принципи. Направен е добър анализ и подбор кой метод в кой от трите класа да бъде имплементирана, за да се постигне най-голяма ефективност.
3. Методите **union** (обединение), **intersection** (сечение), **concatenation** (конкатенация), **iteration** (звезда на Клини) и **addition** (допълнение) са имплементирани, следвайки оптимални алгоритми, взети от учебника по ЕАИ на Стефан Вътев. В класа DFA (детерминиран краен автомат) има 2 различни реализации на обединение – единият подход ни връща автомат от вида DFA (детерминиран краен автомат), който би трябвало да е NFA, но за удобство при употреба с други методи взех решение връщаният резултат да е от тип DFA&. Другият подход за операцията обединение следва една важна идея от курса по ЕАИ – използването на състояния, които са от тип pair<string, string>, за разлика от останалите случаи, в които те са от тип string. Така получаваме PDFA (детерминиран краен автомат със състояния от тип наредена двойка), който всъщност е и третият вид автомат, който използваме за реализацията на библиотеката.
4. Освен основните няколко задължителни конструктори, деструктор и оператор=. В библиотеката са имплементирани гетъри и сетъри за всеки клас. Още повече, създаден е и конструктор на DFA, който го създава по даден PDFA, както и метод DFA::toNFA(), който приема DFA и го превръща в NFA.
5. Проектиран е ефективен начин за четене от файл и дисплейване на резултати на екрана, както се изисква в заданието, чрез метода processInput.
6. Създаден е клас RegEx, който се отнася за регулярния израз, който получаваме на входа под формата на string. Чрез метода RegEx::transform() той се превръща в DFA.
7. Използвани са следните вградени структури от данни : map, set, stack, queue, vector. Освен това сме си създали и клас Alphabet за азбуката на автомата.
8. **Реализация и тестване**

Продуктът е тестван и изпълнява зададената функционалност. Освен това е направена валидация на данните и не се допуска въвеждането на грешни данни, които могат да създадат проблеми. Предотвратено е въвеждането на празни данни, подаване на невалиден брой параметри и промяна на несъществуващи данни. Програмата се изпълнява по следния начин: въвежда се string regex, който се използва като аргумент в подходящ конструктор за RegEx. После обектът от клас RegEx се трансформира в DFA чрез метода transform. Текстовият файл се валидира и се взимат думите от него, минават през автомата на проверка и се извеждат тези, които могат се разпознават от него.

1. **Заключение**

Библиотеката предразполага клиента към лесна и приятна работа. Възникването на затруднения и пречки е сведено до минимум и са спазени принципите на обектно-ориентираното програмиране. По този начин клиентът е защитен и може свободно да се възползва от цялата функционалност, която продуктът предлага. Методите са реализирани оптимално чрез подходящи структури от данни и изпълняват всяка нужда да потребителя по най-добрия начин.