Документация на проект по Обектно-ориентирано програмиране

Глава 1. Увод

1.1. Описание на идеята на проекта

Проектът представя краен недетерминиран автомат с функционалности за детерминизация, минимализация, тотализация, проверка дали дадена дума се разпознава от автомата, проверка дали автоматът приема празната дума, обединение на два автомата, конкатенация на два автомата, звезда на Клини на автомат, съставяне на автомат по даден регулярен израз и съставяне на регулярен израз по даден автомат.

1.2. Цел и задачи на разработката

Основната цел на проекта е да се демонострират усвоени знания в областта на обектно-ориентираното програмиране и владеене на основните парадигми. Друга цел е да се затвърдят знанията за крайни автомати, придобити в курса по Езици, автомати, изчислимост.

Важна задача в разработката е пресъздаването на математически модел на сметачна машина (математически обект) чрез парадигмите на обектно-ориентираното програмиране, така че да бъде възможна работа на компютър с математическия обект, а именно неговото моделиране.

1.3.

В настоящата документация са описани факти относно предметната област на проекта като основни дефиниции и алгоритми. Разглеждат се решенията на проблеми с различна сложност и различни подходи. Описана е общата архитектура и аргументи за избора ѝ. Също така могат да се намерят диаграми, представящи структурата.

Описани са и важни моменти от реализацията на отделните съставни части, на които се разбива проекта. Обърнато е внимание на управлението на паметта и реализцаията на алгоритмите. Разглеждат се направените оптимизации. Описани са тестови сценарии заедно с идеята и проблемът, който проверяват.

Глава 2. Преглед на предметната област

2.1. Основни дефиниции и алгоритми

2.1.1. Дефиниции:

Казваме, че $\mathcal{A}$ разпознава езика L, ако $\mathcal{A}$ разпознава точно думите от L, т.е. $L = \{\alpha \in \Sigma^* \mid \mathcal{A}$ разпознава $\alpha\}$.

Недетерминистичен краен автомат представлява $\mathcal{N} = \langle \Sigma, Q, q_{start}, \Delta, F \rangle$,

- Q е крайно множество от състояния

-Σ е крайна азбука

- $\Delta : Q \times \Sigma \to P(Q)$ е функцията на преходите. Да обърнем внимание, че е възможно за някоя двойка (q, a) да няма нито един преход в автомата. Това е възможно, когато $\Delta(q, a) = \varnothing$

- $q_{start} \in Q$ е началното състояние

-$F \subseteq Q$ е множеството от финални състояния.

2.1.2. Теореми:

Теорема (Рабин-Скот): За всеки недетерминиран краен автомат $\mathcal{N}$ съществува еквивалентен на него детерминиран краен автомат $\mathcal{D}$, т.е. $L(\mathcal{N}) = L(\mathcal{D})$.

Теорема (Клини): Всеки автоматен език се описва с регулярен израз.

2.1.3. Алгоритми:

Алгоритмите са показани на упражненията по Езици, автомати, изчислимост.

Алгоритъм за обединение на два автомата: Обединение на два автомата получаваме, като обединим началните им състояние т.е. като добавим преход от стартовото състояние на първия (или втория) към следстартовото състояние на втория (или първия) със съответната буква.

Алгоритъм за конкатенация на два автомата: Конкатенация на два автомата получаваме, като добъвим преходи от всички финални състояния на първия автомат към всички следстартови сътояния на втория автомат със съответните букви.

Алгоритъм за звезда на Клини на автомат: Звезда на Клини на автомат получаваме като от всички финални състояния на автомата добавим преход към следстартовите му състояния със съответната буква и освен това добавим ново стартово състояние, направим го финално и го свържем със следстартовите състояния със съответната буква.

Алгоритъм за тотализация: Тотализираме автомат, като добавим ново състояние за грешка и от всички състояние, към които няма преход с която и да е буква от азбуката, добавим такъв към новото състояние. Също трябва да се добавят преходи от състоянието за грешка до самото него с всички букви от азбуката.

2.2. Дефиниране на проблеми и сложност на поставената задача

Съществен проблем на проекта е представянето на автомата в компютърната памет и изпълнението на алгоритмите върху него без подходящи структури от данни. Изисква се абстрактно мислене и

разбиране на алгоритмите до такава степен, че да може да бъдат преведени на *C++* с наличните техники и подходи.

2.3. Подходи и методи за решаване на проблемите

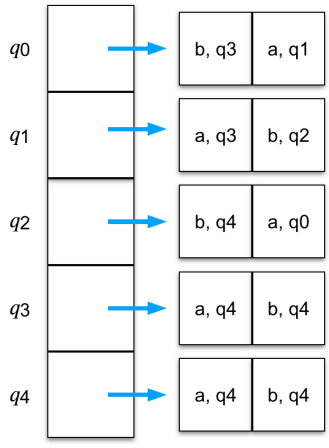
Печеливш подход за решаване на възникнали проблеми е консултация с по-опитни колеги и задълбочен прочит и осмисляне на алгоритмите.

Глава 3. Проектиране

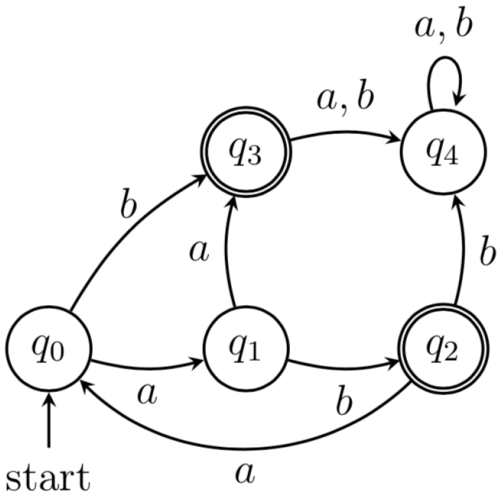
3.1. Обща архитектура

На математическо ниво автоматът се представя като граф - върховете на графа отговарят на състоянията на автомата, а ребрата представят преходите. Графът е представен в компютърната памет чрез списък на съседство - за всеки връх е даден списък, в който са изредени неговите съседи. Основен градивен елемент на проекта е шаблонен динамичен масив. Списъкът на съседство е имплементиран чрез динамичен масив от динамични масиви от специален клас “преход”, в който има две член-данни - буква и връх (състояние, в което се отива с тази буква). Използван е динамичен масив за запазване на финалните състояния. За удобство за работа със символни низове е разписана имплементация на класа *string.*

3.2. Диаграма



Представяне на автомата в паметта



Автомат

Глава 4. Реализация и тестване

4.1. Реализация на класове

Реализация на шаблонен клас динамичен масив:

```
template <typename T>
class DynamicArray
{
public:
    DynamicArray();
    DynamicArray(const DynamicArray<T>&);
    DynamicArray& operator=(const DynamicArray&);
    const T& operator[](unsigned) const;
    T& operator[](unsigned);
    bool operator==(const DynamicArray<T>&) const;
    ~DynamicArray();

public:
    void addElement(const T&);
    void removeElement(const T&);
    int findElementIndex(const T&) const;
    unsigned getSize() const;

private:
    void free();
    void resize();
    void copy(const DynamicArray<T>&);
    int linearSearch(const T&) const;

private:
    T* dynamicArray;
    unsigned size;
    unsigned capacity;
};
```

Реализация на клас “преход”:

```
class Transition
{
public:
    Transition() = default;
    Transition(unsigned, char);
    unsigned getToVertex() const;
    char getLetter() const;

private:
    unsigned toVertex;
    char letter;
};
```

Реализация на клас автомат:

```
class Automaton
{
public:
    Automaton();
    Automaton& unionOf2(const Automaton&) const;
```

```
    Automaton& concatenationOf2(const Automaton&) const;
    Automaton& kleeneStar() const;
    void determine();
    void totalize();


private:
    void addTransition(unsigned fromVertex, unsigned toVertex, char letter);
    void copyTransitionsForUnion(const Automaton&, const Automaton&);
    void copyTransitionsForConcatenation(const Automaton&, const Automaton&);

private:
   DynamicArray<DynamicArray<Transition>>  automaton;
    DynamicArray<unsigned> finalStates;
};
```

4.2. Управление на паметта и реализация на алгоритмите

В проекта е спазен подходът всеки обект да се грижи за паметта си. В класовете, които използват динамична памет, е разписано съответното канонично представяне.

Алгоритмите са строго специфични и подлежат на оптимизации. Често сложността им е $\Theta(n^3)$ или по-висока.

4.3. Описание на тестови сценарии

Примерът представя автомата от схемата по-горе - реализира се звезда на Клини, автоматът се тотализира, проверява се дали приема два символни низа и също така дали приема празната дума.

Глава 7. Заключение

7.1. Обобщение на изпълнението на началните цели

Съществената част е изпълнена, а именно затвърждаване на знанията по обектно-ориентирано програмиране. Не всички алгоритми са реализирани поради сложността на материята - след консултация с по-опитни колеги ще бъдат добавени.

7.2. Насоки за бъдещо развитие и усъвършенстване

Проектът подлежи на сериозна оптимизация по отношение на алгоритмите за модификация на автомата. Добре е да се поправи според мнението на по-опитни колеги и преподаватели.

Използвана литература

Записки по “Езици, автомати, изчислимост” на Стефан Вътев

Записки от упражнения по Езици, автомати, изчислимост

Уикипедия - статия за краен автомат