ТЕМА 12: РЕГУЛЯРНИ ИЗРАЗИ

Регулярни изрази са множества от низове, които дефинират шаблони за търсене. Правилата се задават с оператори като:

- съединяване (конкатенация) xy;
- алтернатива x|y (x или y, записва се още като x+y);
- повторение x* (x се повтаря 0 или повече пъти);
- x? (1 или повече пъти) и др.

Регулярните изрази се дефинират формално чрез следните рекурсивни правила:

1) Всеки символ от азбуката на формален език е регулярен израз;

2) Празният низ ε е регулярен израз;

3) Ако R1 и R2 са регулярни изрази, то (R1), (R2), R1R2, R1 | R2, R1 * , R2 * са също регулярни изрази;

4) Никакъв друг израз не е регулярен.

Чрез регулярни изрази мога да бъдат дефинирани т.нар. лексеми в програмните езици – ключови думи, идентификатори, оператори, разделители, константни.

Например регулярният израз (0|1|2|3|4|5|6|7|8|9)+ дефинира целочислена константа, т.е. цяло число, съставено от една или повече цифри. Символът '+' е специален символ в регулярния израз, който означава едно или повече повторения.

Регулярен е език, който се поражда с регулярна граматика.

Регулярни изрази могат да се използват за търсене на различни последователности от символи в текст.

В информатиката регулярен израз (на английски: *regular expression*, съкращавано понякога като regex или regexp) е последователност от знаци, която дефинира шаблон за търсене. Обикновено този шаблон се използва от алгоритми за претърсване на низове за операции от типа „търсене“ или „търсене и заместване“ върху низове или за проверка на валидността на въведени данни. Това е техника, разработена в рамките на теоретичната информатика и теорията на формалните езици.

Регулярните изрази се използват в търсачките, диалозите за търсене и заместване в текстообработващите програми и текстовите редактори, в текстообработващи инструменти като sed и AWK и при лексикален анализ. Много езици за програмиране предоставят възможности за работа с регулярни изрази, вградени или достъпни чрез библиотеки.

Понятието възниква през 50-те години, когато американският математик Стивън Коул Клийни формализира описанието на *регулярните езици*. То влиза в широка употреба във връзка със средствата за текстообработка на Unix. От 80-те години съществуват различни синтаксиси за писане на регулярни изрази. Един от тях е на POSIX, а друг, широко използван – на Perl.

Регулярните изрази, често наричани на английски и *patterns* (шаблони), служат за задаване на множества от низове, необходими за определена цел. Един прост начин да се зададе крайно множество от низове е да се изброят елементите му. Често обаче има по-компактен начин да се опише желаното множество. Например множеството от трите низа "Handel", "Händel" и "Haendel" може да бъде описано с шаблона H(ä|ae?)ndel; казваме, че този шаблон **съответства на** или **съвпада с** всеки от трите низа. В повечето формализми, ако на дадено множество съответства поне един регулярен израз, то съществуват и безкрайно количество други регулярни изрази, които също му съответстват – описанието не е уникално. Повечето формализми предоставят следните операции за конструиране на регулярни изрази.

Логическо „или“

С вертикална черта се разделят алтернативи. Например gray|grey съвпада с "gray" и "grey".

Групиране

Скобите се използват за задаване на област на действие и приоритет на операторите (както и за други цели). Например, gray|grey и gr(a|e)y са еквивалентни шаблони, които описват множеството от двата низа "gray" и "grey".

Квантори

Квантор, добавен след даден синтактичен елемент (например знак) или група указва колко срещания на елемента се допускат. Най-често употребяваните квантори са въпросителен знак ?, звездичка * (от операцията звезда на Клийни), и знак плюс + (плюс на Клийни).

? Въпросителният знак означава *нула или едно* срещане на предходния елемент. Например colou?r съвпада с "color" и "colour".

***** Звездичката означава *нула или повече* срещания на предходния елемент. Например ab*c съвпада с "ac", "abc", "abbc", "abbbc" и т.н.

+ Знакът плюс означава *едно или повече* срещания на предходния елемент. Например ab+c съвпада с "abc", "abbc", "abbbc" и т.н., но не и с "ac".

{n}[15] Допускат се точно *n* поредни съответствия на предходния елемент.

{min,}[15] Допускат се *min* или повече поредни съответствия на предходния елемент.

{min,max}[15] Допускат се поне *min* но не повече от *max* поредни съответствия на предходния елемент.

Заместител

Заместващият символ **.** съвпада с произволен знак. Например a.b съвпада с всеки низ, който съдържа "a", последван от произволен знак и след това "b", а a.*b съответства на всеки низ, който съдържа "a", последвано след някакъв брой знаци от "b".

Тези конструкции могат да бъдат комбинирани, за да се изграждат произволно сложни изрази, точно както аритметичните изрази могат да се конструират от числа и операциите +, −, **.** и **:**. Например, както H(ae?|ä)ndel, така и H(a|ae|ä)ndel са валидни шаблони, които съответстват на същите низове като в предходния пример, H(ä|ae?)ndel.

Точният синтаксис на регулярните изрази варира според софтуера и контекста.

Примери за приложение на регулярни изрази

1. **Да се опишат регулярните изрази над азбуката A={0,1}**

   01*=0(1)* ={0,01,011,0111,….}- 0 следвана от 1или няколко 1;

   (01*)(01) ={001,0101,01101,011101,….}- 0 следвана от 1или няколко 1, следвани от 01;

   0+1={0,1}- думи с дължина 1;

   (0+1)*={ε,0,1,00,01,10,11…}- всички думи, ε – празна дума;

   (0+1)*010 – думи, завършващи на 010;

   (0+1)*01(0+1)* - всички думи, съдържащи 01

2. **Да се напише регулярен израз над азбуката A={0,1}, който разпознава думи с нечетна дължина, и 0 и 1 се редуват.**

   0(10)* - дума, започваща с 0;

   1(01)* - дума, започваща с 1;

   Регулярният израз е:

   0(10)*+ 1(01)*

3. **Да се напише регулярен израз над азбуката A={0,1}, който разпознава думи, несъдържащи две последователни 1.**

   Дума не съдържа 1 – 0*

   Дума само с една 1 - 0*10*

   Дума започва с 1 – 1(00*1)*0*

   Дума завършва на 1 – 0*(10*0)*1

   Дума започва и завършва с 0 или 1 и съдържа поне една 1 – 0*(10*0)*1(00*1)*0*

   Регулярният израз може да се запише по следните начини:

   1. 0*(10*0)*1(00*1)*0* + 0*
   2. 0*(10*0)*(1+ ε)(00*1)*0* + 0*