Регулярные выражения

Основные определения

Регулярные выражения в алфавите Σ и регулярные множества, которые они обозначают, определяются рекурсивно следующим образом:

1) $\varnothing$ – регулярное выражение, обозначающее регулярное множество $\varnothing$;

2) e – регулярное выражение, обозначающее регулярное множество $\{e\}$;

3) если $a \in \Sigma$, то a – регулярное выражение, обозначающее регулярное множество $\{a\}$;

4) если p и q – регулярные выражения, обозначающие регулярные множества P и Q, то

   а) $(p+q)$ – регулярное выражение, обозначающее $P \cup Q$;

   б) pq – регулярное выражение, обозначающее PQ;

   в) p^* – регулярное выражение, обозначающее P^*;

5) ничто другое не является регулярным выражением.

Основные определения

Расстановка приоритетов:

- * (итерация) – наивысший приоритет;
- конкатенация;
- + (объединение).

Таким образом, $0 + 10^* = (0 + (1\ (0^*)))$. Примеры:

1. 01 означает {01};
2. 0^* – $\{0^*\}$;
3. $(0+1)^*$ – $\{0, 1\}^*$;
4. $(0+1)^*$ 011 – означает множество всех цепочек, составленных из 0 и 1 и оканчивающихся цепочкой 011;
5. $(a+b)\ (a+b+0+1)^*$ означает множество всех цепочек $\{0, 1, a, b\}^*$, начинающихся с a или b.

Основные определения

Леммы:

1) $\alpha + \beta = \beta + \alpha$
2) $\varnothing^* = e$
3) $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$
4) $\alpha(\beta\gamma) = (\alpha\beta)\gamma$
5) $\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$
6) $(\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma$
7) $\alpha e = e\alpha = \alpha$
8) $\alpha\varnothing = \varnothing\alpha = \varnothing$
9) $\alpha+\alpha^* = \alpha^*$
10) $(\alpha^*)^* = \alpha^*$
11) $\alpha+\alpha = \alpha$
12) $\alpha+\varnothing = \alpha$

Связь РВ и РМ

РМ – языки, порождаемые РВ. Например:

$$x = a+b,\ y = c+d,$$

$$x \Leftrightarrow X = \{a, b\},\ y \Leftrightarrow Y = \{c, d\},$$

$$x + y \Leftrightarrow X \cup Y = \{a, b, c, d\}.$$

Конкатенация:

$$xy \Leftrightarrow XY = \{ac, ad, bc, bd\}.$$

$$к(и+о)т \Leftrightarrow \{к\}\{и, о\}\{т\} = \{кит, кот\}$$

или по леммам №5 и №6

$$к(и+о)т = кит + кот \Leftrightarrow \{кит, кот\}.$$

Итерация:

$$x = a,$$

$$x^* \Leftrightarrow X^* = \{e, a, aa, aaa, \ldots\},\ \text{т.е.}$$

$$x^* = e + x + xx + xxx + \ldots$$

Связь РВ и РМ

Итерация конкатенации и объединения:

$$(xy)^* = e + xy + xyxy + xyxyxy + \ldots$$

$$(x + y)^* = e + (x + y) + (x + y)(x + y) + (x + y)(x + y)(x + y) + \ldots =$$

$$= e + x + xx + xy + yx + yy + xxx + \ldots$$

Пример:

$0 + 1(0+1)^* \Leftrightarrow \{0\} \cup (\{1\} \cup \{0, 1\}^*) = \{0, 1, 10, 11, 100, 101, 110, 111\ldots\}.$

Объединение коммутативно: $x + y = y + x$

Конкатенация – нет: $xy \neq yx$

Связь РВ и РМ

Примеры на приоритет:

$$x + yz \Leftrightarrow \{x, yz\},$$

$$(x + y)z \Leftrightarrow \{xz, yz\},$$

$$x + y^* \Leftrightarrow \{e, x, y, yy, yyy, yyyy, \ldots\},$$

$$(x + y)^* \Leftrightarrow \{e, x, y, xx, xy, yx, yy, xxx, \ldots\},$$

$$(xy)^* \Leftrightarrow \{e, xy, xyxy, \ldots\},$$

$$xy^* \Leftrightarrow \{x, xy, xyy, xyyy, \ldots\}.$$

Новые леммы:

- $a^* + e = a^*$;
- $(a + e)^* = a^*$;
- $a^* a^* = a^*$;
- $e^* = e$;
- и т.д.

Регулярные системы уравнений

Уравнение с регулярными коэффициентами

$$X = aX + b$$

имеет решение (наименьшую неподвижную точку) a^*b:

$$aa^*b + b = (aa^* + e)b = a^*b$$

Система уравнений с регулярными коэффициентами:

$$X_1 = \alpha_{10} + \alpha_{11}X_1 + \alpha_{12}X_2 + \ldots + \alpha_{1n}X_n$$

$$X_2 = \alpha_{20} + \alpha_{21}X_1 + \alpha_{22}X_2 + \ldots + \alpha_{2n}X_n$$

…………………………………………………..

$$X_n = \alpha_{n0} + \alpha_{n1}X_1 + \alpha_{n2}X_2 + \ldots + \alpha_{nn}X_n$$

Неизвестные – $\Delta = \{X_1, X_2, \ldots, X_n\}$.

Регулярные системы уравнений

Алгоритм решения (метод Гаусса):

Шаг 1. Положить $i = 1$.

Шаг 2. Если $i = n$, перейти к шагу 4. Иначе записать уравнения для X_i в виде $X_i = \alpha X_i + \beta$ ($\beta = \beta_0 + \beta_{i+1} X_{i+1} + \ldots + \beta_n X_n$). Затем в правых частях для уравнений $X_{i+1}, \ldots, X_n$ заменим X_i регулярным выражением $\alpha^* \beta$.

Шаг 3. Увеличить i на 1 и вернуться к шагу 2.

Шаг 4. Записать уравнение для X_n в виде $X_n = \alpha X_n + \beta$. Перейти к шагу 5 (при этом $i = n$).

Шаг 5. Уравнение для X_i имеет вид $X_i = \alpha X_i + \beta$. Записать на выходе $X_i = \alpha^* \beta$, в уравнениях для $X_{i-1}, \ldots, X_1$ подставляя $\alpha^* \beta$ вместо X_i.

Шаг 6. Если $i = 1$, остановиться, в противном случае уменьшить i на 1 и вернуться к шагу 5.

Преобразование ДКА в РВ

Для ДКА $M = (Q, \Sigma, \delta, q_0, F)$ составим систему с регулярными коэффициентами где $\Delta = Q$:

1. полагаем $\alpha_{ij} := \varnothing$;
2. если $\delta(X_i, a) = X_j$, $a \in \Sigma$, то $\alpha_{ij} := \alpha_{ij} + a$;
3. если $X_i \in F$ или $\delta(X_i, \perp) = HALT$, то $\alpha_{i0} := e$.

После решения искомое РВ будет $X_1 = q_0$.

Преобразование ДКА в РВ

Пример: для числа с фиксированной точкой получим систему

$$q_0 = \varnothing + \varnothing q_0 + sq_1 + pq_2 + dq_3 + \varnothing q_4$$
$$q_1 = \varnothing + \varnothing q_0 + \varnothing q_1 + pq_2 + dq_3 + \varnothing q_4$$
$$q_2 = \varnothing + \varnothing q_0 + \varnothing q_1 + \varnothing q_2 + \varnothing q_3 + dq_4$$
$$q_3 = e + \varnothing q_0 + \varnothing q_1 + \varnothing q_2 + dq_3 + pq_4$$
$$q_4 = e + \varnothing q_0 + \varnothing q_1 + \varnothing q_2 + \varnothing q_3 + dq_4$$

Здесь:

- s – знак числа, s = '+' + '–';
- p – десятичная точка, p = '.';
- d – цифры, d = '0' + '1' + … + '9'.

Преобразование ДКА в РВ

Решение:

$q_0 = \varnothing^*(sq_1 + pq_2 + dq_3 + \varnothing q_4 + \varnothing) = sq_1 + pq_2 + dq_3 \Rightarrow$

$q_1 = \varnothing + \varnothing q_0 + \varnothing q_1 + pq_2 + dq_3 + \varnothing q_4 = pq_2 + dq_3,$

$q_2 = \varnothing + \varnothing q_0 + \varnothing q_1 + \varnothing q_2 + \varnothing q_3 + dq_4 = dq_4,$

$q_3 = e + \varnothing q_0 + \varnothing q_1 + \varnothing q_2 + dq_3 + pq_4 = dq_3 + pq_4 + e,$

$q_4 = e + \varnothing q_0 + \varnothing q_1 + \varnothing q_2 + \varnothing q_3 + dq_4 = dq_4 + e.$

Из третьего уравнения:

$q_3 = dq_3 + pq_4 + e = d^*(pq_4 + e).$

Из четвертого уравнения:

$q_4 = dq_4 + e = d^*e = d^*.$

Преобразование ДКА в РВ

Обратный ход:

$q_3 = d^*(pq_4 + e) = d^*(pd^* + e)$,

$q_2 = dq_4 = dd^*$,

$q_1 = pq_2 + dq_3 = pdd^* + dd^*(pd^* + e)$,

$q_0 = sq_1 + pq_2 + dq_3 = s(pdd^* + dd^*(pd^* + e)) + pdd^* + dd^*(pd^* + e)$.

Таким образом, данному ДКА соответствует РВ

$$s(pdd^* + dd^*(pd^* + e)) + pdd^* + dd^*(pd^* + e).$$

Упростим:

$$s(pdd^* + dd^*(pd^* + e)) + pdd^* + dd^*(pd^* + e) =$$

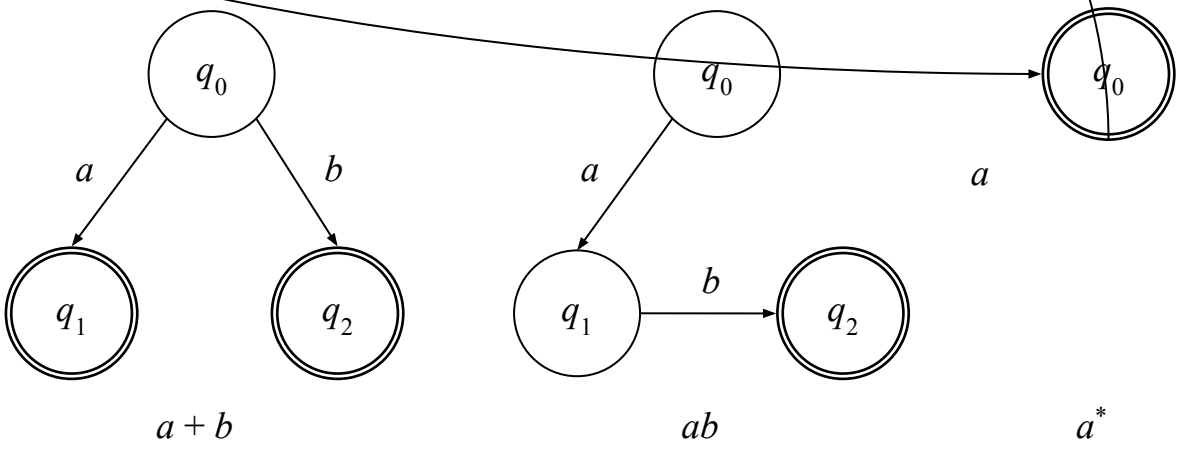
$$= spdd^* + sdd^*(pd^* + e) + pdd^* + dd^*(pd^* + e) = (s + e)(pdd^* + dd^*(pd^* + e))$$

Для более короткой записи можно использовать положительную итерацию $aa^* = a^*a = a^+$:

$$(s + e)(pdd^* + dd^*(pd^* + e)) = (s + e)(pd^+ + d^+(pd^* + e)) = (s + e)(pd^+ + d^+pd^* + d^+)$$

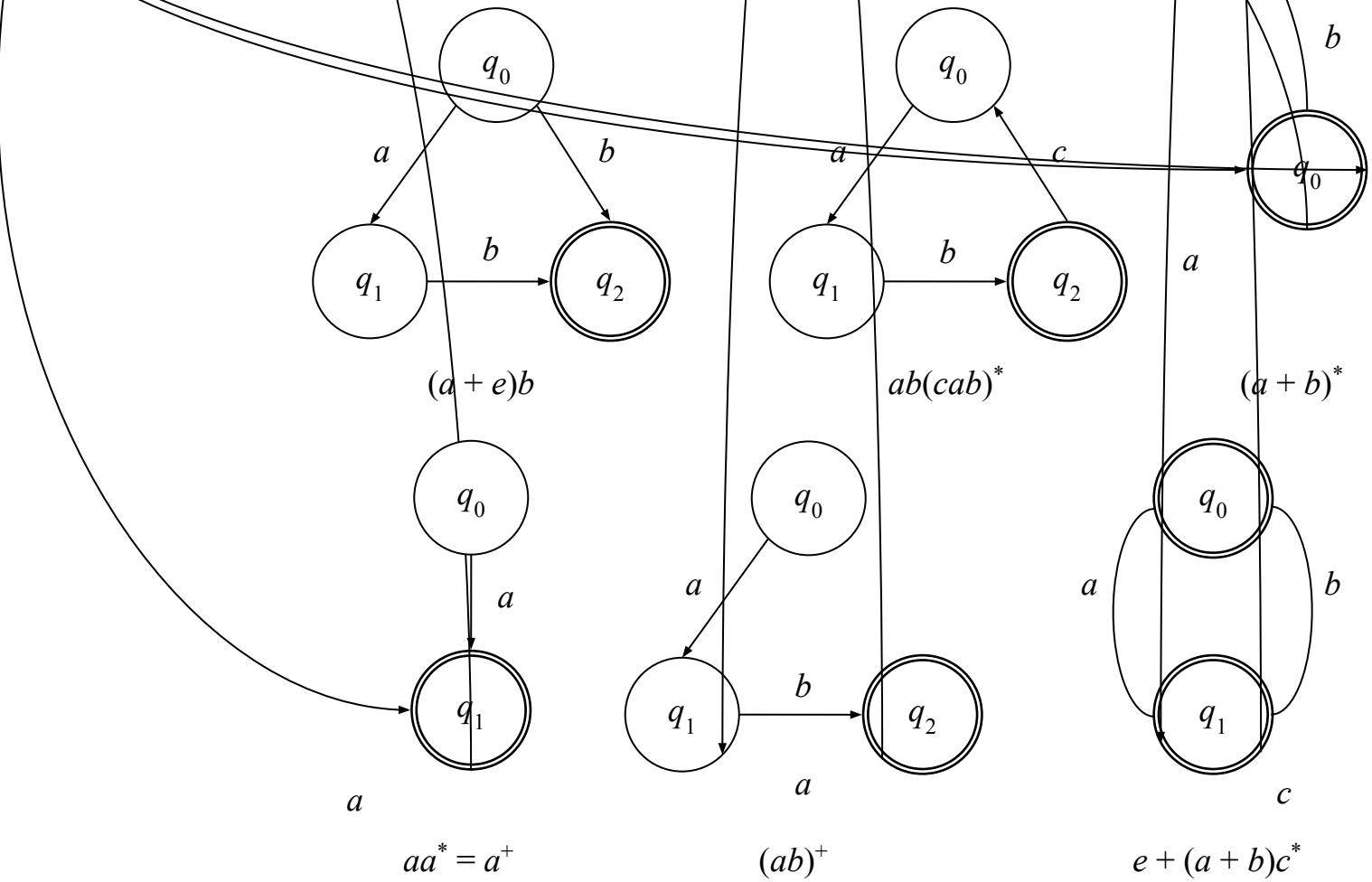
Преобразование ДКА в РВ

Сопоставление графа функции переходов ДКА основным операциям с регулярными выражениями:



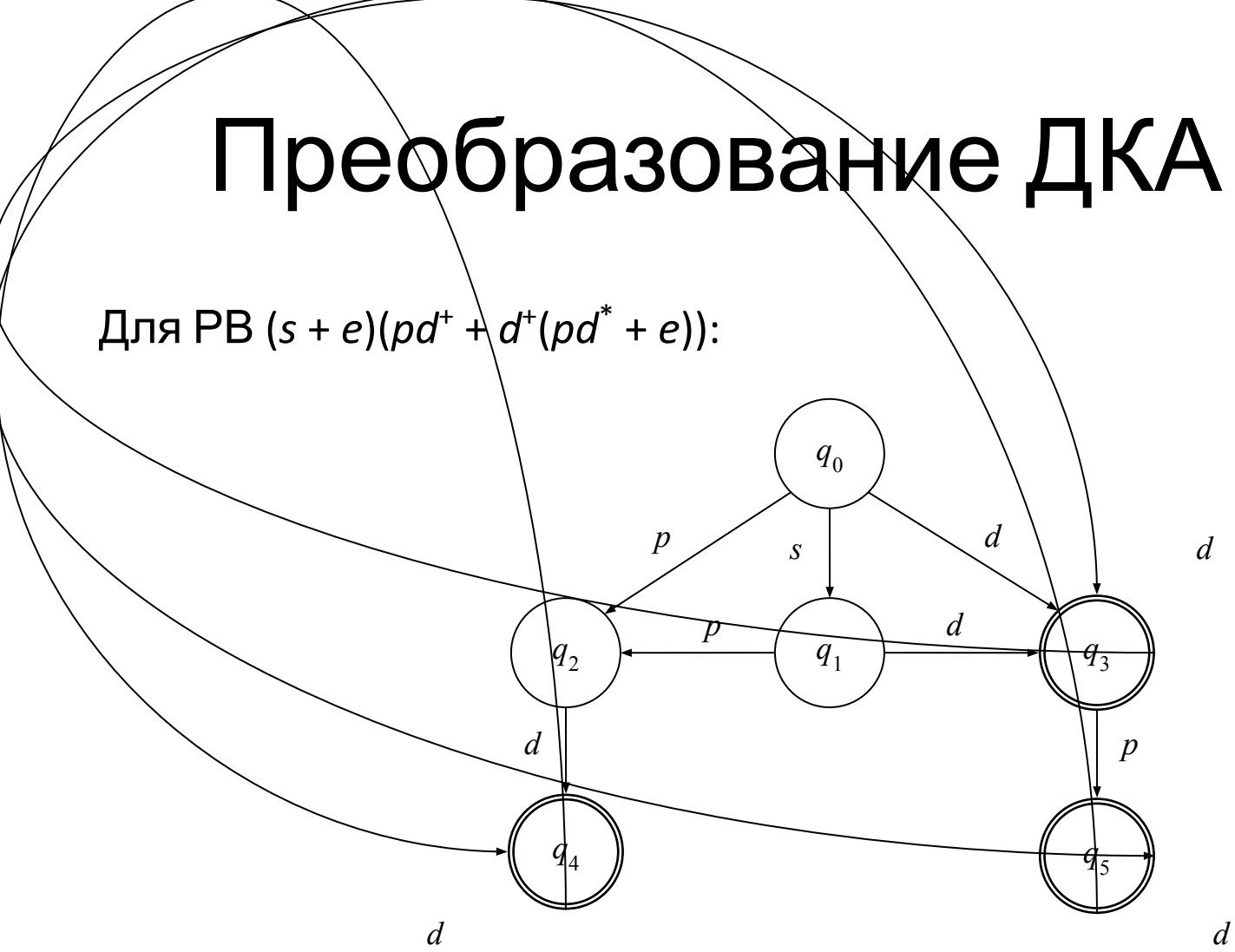
Преобразование ДКА в РВ

Более сложные комбинации операций:



Преобразование ДКА в РВ

Для РВ $(s + e)(pd^{+} + d^{+}(pd^{*} + e))$:



Программирование РВ

Регулярные выражения:

- Встроены во многие языки программирования (PHP, JavaScript, …);
- Реализованы в виде подключаемых компонентов (например, класс Regex для платформы .NET).

Отличия в формах записи:

$$x? = x + e$$

$$x\{1,3\} = x + xx + xxx$$

и т.д.

Программирование РВ

Конструкции класса Regex (System.Text.RegularExpressions):

Символ	Интерпретация
Escape-последовательности	
\b	При использовании его в квадратных скобках соответствует символу «←» (\u0008)
\t, \r, \n, \a, \f, \v	Табуляция (\u0009), возврат каретки (\u000D), новая строка (\u000A) и т.д.
\cX	Управляющий символ (например, \cC – это Ctrl+C, \u0003)
\e	Escape (\u001B)
\ooo	Символ ASCII в восьмеричной системе
\xhh	Символ ASCII в шестнадцатеричной системе
\uhhhh	Символ Unicode
\	Следующий символ не является специальным символом РВ. Этим символом нужно экранировать все специальные символы
Пример (в примере приведен шаблон и строка поиска, в строке найденные совпадения подчеркнуты): @"\r\n\w+" – "\r\nЗдесь имеются\nдве строки".	

Программирование РВ

Подмножества символов	
.	Любой символ, кроме конца строки (\n)
[xxx]	Любой символ из множества
[^xxx]	Любой символ, кроме символов из множества
[x-x]	Любой символ из диапазона
[xxx-[xxx]]	Вычитание одного множества или диапазона из другого
\p{name}	Любой символ, заданный категорией Unicode с именем name
\P{name}	Любой символ, кроме заданных категорией Unicode с именем name
\w	Множество символов, используемых при задании идентификаторов
\W	Множество символов, не используемых при задании идентификаторов
\s	Пробелы
\S	Все, кроме пробелов
\d	Цифры
\D	Не цифры

Примеры:

@".+" – "\r\nЗдесь имеются\nдве строки"; @"[fx]+" – "0xabcfx";

@"[^fx]+" – "0xabcfx"; @"[a-f]+" – "0xabcfx";

@"[^a-f]+" – "0xabcfx"; @"[a-z-[c]]+" – "0xabcfx";

@"\p{Lu}" – "City Lights"; // Lu – прописные буквы

@"\P{Lu}" – "city";

@"\p{IsCyrillic}" – "хаОС"; // IsCyrillic – русские буквы

Программирование РВ

Привязка	
^, \A	В начале строки
$, \Z	В конце строки или до символа «\n» в конце строки
\z	В конце строки
\G	В том месте, где заканчивается предыдущее соответствие
\b	Граница слова
\B	Любая позиция не на границе слова

Примеры:

@"\G\(\d\)" – "(1)(3)(5)[7](9) "; // три соответствия (1), (2) и (3)

@"\bn\S*ion\b" – "nation donation";

@"\Bend\w*\b" – "end sends endure lender".

Программирование РВ

Операции (кванторы)	
*, *?	Итерация
+, +?	Положительная итерация
?, ??	Ноль или одно соответствие
{n}, {n}?	Точно n соответствий
{n,}, {n,}?	По меньшей мере, n соответствий
{n,m},{n,m}?	От n до m соответствий

Примеры (первые кванторы – жадные, ищут как можно большее число элементов, вторые – ленивые, ищут как можно меньшее число элементов):

@"\d{3,}" – "888-555-5555";

@"^\d{3}" – "913-913-913";

@"-\d{3}$" – "913-913-913";

@"5+?5" – "888-555-5555"; // три совпадения – 55, 55 и 55

@"5+5" – "888-555-5555".

Программирование РВ

Группирование	
()	Группа, автоматически получающая номер
(?:)	Не сохранять группу
(?<имя>) или (?'имя')	При обнаружении соответствия создается именованная группа
(?<имя–имя>) или (?'имя– имя')	Удаление ранее определенной группы и сохранение в новой группе подстроки между ранее определенной группой и новой группой
(?imnsx:) (?–imnsx:)	Включает или выключает в группе любую из пяти возможных опций: i – нечувствительность к регистру; s – одна строка (тогда «.» – это любой символ); m – многострочный режим («^», «$» – начало и конец каждой строки); n – не захватывать неименованные группы; x – исключить не преобразованные в escape-последовательность пробелы из шаблона и включить комментарии после знака номера (#)

Программирование РВ

(?!)	Отрицательное утверждение просмотра вперед нулевой длины
(?<=)	Положительное утверждение просмотра назад нулевой длины
(?<!)	Отрицательное утверждение просмотра назад нулевой длины
(?>)	Невозвращаемая (жадная) часть выражения

Примеры:

@"(an)+" – "bananas annals";

@"an+" – "bananas annals"; // сравните, три совпадения – an, an и ann

@"(?i:an)+" – "baNAnas annals";

@"[a-z]+(?=\d)" – "abc xyz12 555w";

@"(?<=\d)[a-z]+" – "abc xyz12 555w".

Программирование РВ

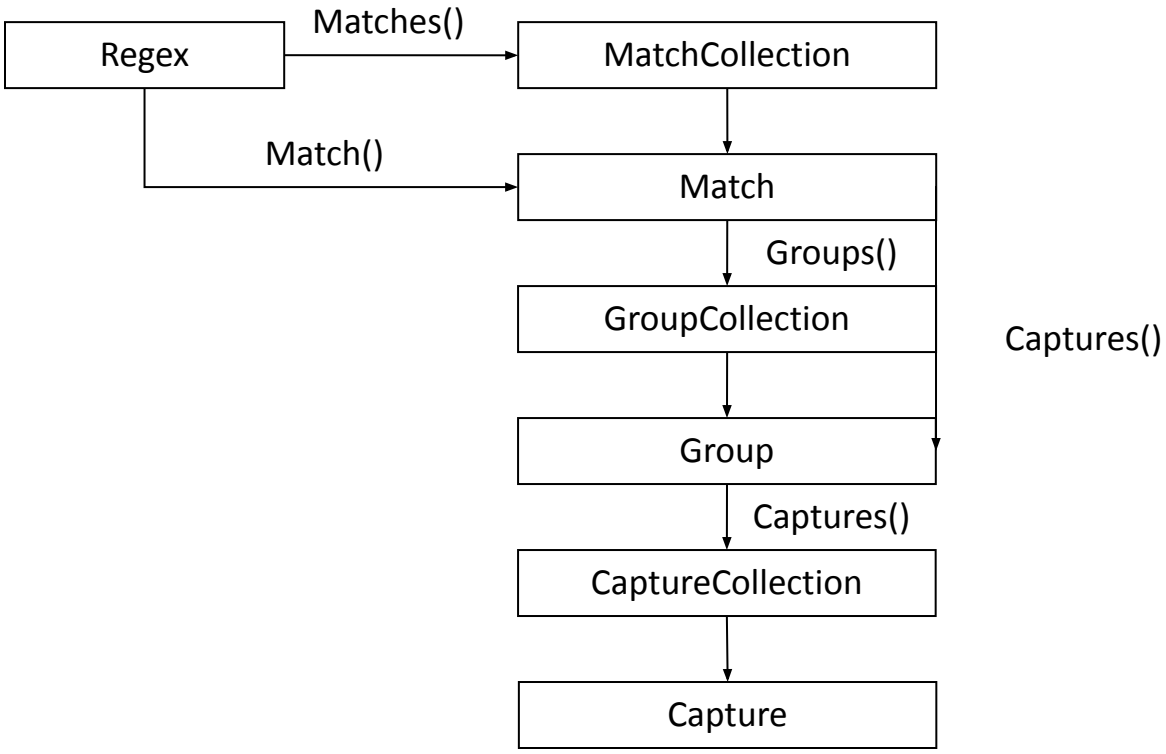
Ссылки	
\число	Ссылка на группу
\k<имя>	Ссылка на именованную группу
Примеры: @"(\w)\1" – "deep"; @"(?<char>\w)\k<char>" – "deep".	
Конструкции изменения	
|	Альтернатива (соответствует операции объединения)
(?(выражение) да|нет)	Сопоставляется с частью «да», если выражение соответствует; в противном случае сопоставляется с необязательной частью «нет»
(?(имя)да|нет), (?(число)да|нет)	Сопоставляется с частью «да», если названное имя захвата имеет соответствие; в противном случае сопоставляется с необязательной частью «нет»
Пример: @"th(e|is|at)" – "this is the day";	

Программирование PB

Подстановки	
$число	Замещается часть строки, соответствующая группе с указанным номером
${имя}	Замещается часть строки, соответствующая группе с указанным именем
$$	Подставляется $
$&	Замещение копией полного соответствия
$`	Замещение текста входной строки до соответствия
$'	Замещение текста входной строки после соответствия
$+	Замещение последней захваченной группы
$_	Замещение всей строки
Комментарии	
(?#)	Встроенный комментарий
#	Комментарий до конца строки

Программирование РВ

Результаты работы Regex:



Программирование РВ

Пример на языке C#:

```
Regex r = new Regex(@"((\d)+)+");
Match m = r.Match("123 456");
int matchCount = 0;
while (m.Success)
{
        Console.WriteLine("Соответствие {0}", ++matchCount);
        for (int i = 1; i < m.Groups.Count; i++)
        {
                Group g = m.Groups[i];
                Console.WriteLine("    Группа {0} = '{1}'", i, g.Value);
                for (int j = 0; j < g.Captures.Count; j++)
                {
                        Capture c = g.Captures[j];
                        Console.WriteLine("       Захват {0} = '{1}', позиция = {2},
длина = {3}", j, c, c.Index, c.Length);
                }
        }
        m = m.NextMatch();
}
```

```
Соответствие 1
   Группа 1 = '123'
      Захват 0 = '123', позиция = 0, длина = 3
   Группа 2 = '3'
      Захват 0 = '1', позиция = 0, длина = 1
      Захват 1 = '2', позиция = 1, длина = 1
      Захват 2 = '3', позиция = 2, длина = 1
Соответствие 2
   Группа 1 = '456'
      Захват 0 = '456', позиция = 4, длина = 3
   Группа 2 = '6'
      Захват 0 = '4', позиция = 4, длина = 1
      Захват 1 = '5', позиция = 5, длина = 1
      Захват 2 = '6', позиция = 6, длина = 1
```

Программирование РВ

Пример на языке C++ CLI (Visual C++/CLR/Консольное приложение

```
int main()
{
	Regex ^r = gcnew Regex(L"((\\d)+)+");
	Match ^m = r->Match(L"123 456");
	int matchCount = 0;
	while (m->Success)
	{
		Console::WriteLine(L"Соответствие {0}", ++matchCount);
		for (int i = 1; i < m->Groups->Count; i++)
		{
			Group ^g = m->Groups[i];
			Console::WriteLine(L"  Группа {0} = '{1}'", i, g->Value);
			for (int j = 0; j < g->Captures->Count; j++)
			{
				Capture ^c = g->Captures[j];
				Console::WriteLine(L"    Захват {0} = '{1}', позиция = {2}, длина = {3}", j, c, c->Index, c->Length);
			}
		}
		m = m->NextMatch();
	}
	return 0;
}
```

System::Text::RegularExpressions

Включение действий и поиск ошибок

Ограничение количества значащих цифр в числе:

$$(s + e)(pd^{+} + d^{+}(pd^{*} + e))$$

$$s = \backslash+|-$$

$$p = \backslash.$$

$$d = \backslash d$$

$$s + e = s? = (\backslash+|-)?$$

$$pd^{*} + e = (pd^{*})? = (\backslash.\backslash d*)?$$

$\Rightarrow$ @"(\+|-)?(\.\d+|\d+(\.\d*)?)" или @"^(\+|-)?(\.\d+|\d+(\.\d*)?)$"

```
Regex r = new Regex(@"^(\+|-)?(\.(?'digit'\d)+|(?'digit'\d)+(\.(?'digit'\d)*)?)$");
Match m = r.Match("+1.23456789");
if (m.Success)
{
        Group g = m.Groups["digit"];
        if (g.Captures.Count < 9) Console.WriteLine("OK");
        else Console.WriteLine("Ошибка в позиции {0}: мантисса содержит больше 8 значащих цифр",
g.Captures[8].Index + 1);
}
else Console.WriteLine("Строка не содержит число с фиксированной точкой");
```

Включение действий и поиск ошибок

Определение позиции ошибки:

```
Regex r = new Regex(@"(\+|-)?(\.(?'digit'\d)+|(?'digit'\d)+(\.(?'digit'\d)*)?)");
string str = "+1.2345!678";
Match m = r.Match(str);
if (m.Success)
{
        Group g = m.Groups["digit"];
        if (g.Captures.Count < 9)
        {
                if (m.Index > 0) Console.WriteLine("Ошибка в позиции 1: неожиданный символ '{0}'", str[0]);
                else if (m.Length < str.Length) Console.WriteLine("Ошибка в позиции {0}: неожиданный символ '{1}'",
m.Length + 1, str[m.Length]);
                else Console.WriteLine("OK");
        }
        else Console.WriteLine("Ошибка в позиции {0}: мантисса содержит больше 8 значащих цифр",
g.Captures[8].Index + 1);
}
else Console.WriteLine("Строка не содержит число с фиксированной точкой");
```

- «+1.2345!678» – ошибка в позиции 8;
- «!1.2345678» – ошибка в позиции 1

Включение действий и поиск ошибок

Определение позиции ошибки:

1. первая позиция входной цепочки (1), если первое соответствие не начинается с позиции Index = 0;

2. позиция, следующая за последним соответствием (match.Length + 1), если она не совпадает с последней позицией входной цепочки;

3. позиция первого разрыва между соответствиями (match[i].Index + match[i]. Length + 1), если символ, следующий за предыдущим соответствием, не является первым символом следующего соответствия.

Включение действий и поиск ошибок

```
Regex r = new Regex(@"\w+(\.\w+)*");
string str = "abc.xyz.pqr";
MatchCollection m = r.Matches(str);
if (m.Count == 1 && m[0].Value == str) Console.WriteLine("OK");
else if (m.Count == 0) Console.WriteLine("Ошибка в позиции 1 '{0}'", str[0]);
else
{
        int index = 0;
        for (int i = 0; i < m.Count; i++)
        {
                if (m[i].Index > index) break;
                index = m[i].Index + m[i].Length;
        }
        Console.WriteLine("Ошибка в позиции {0} '{1}'", index + 1, str[index]);
}
```

- «abc.xyz.pqr» – правильно;
- «+abc.xyz.pqr» – ошибка в позиции 1 («+»);
- «abc.xyz.pqr!» – ошибка в позиции 12 («!»);
- «abc.xyz!.pqr» – ошибка в позиции 8 («!»).

Включение действий и поиск ошибок

Но! «abc.xyz.+pqr» – ошибка в позиции 8 («.»).

Новый вариант шаблона:

@"\w+(\.\w+)*(\.(?!$))?"

Проверка:

- «abc.xyz.+pqr» – ошибка в позиции 9 («+»);
- «abc.xyz.pqr.» – ошибка в позиции 12 («.»).

Сбалансированные определения

Сбалансированные определения:

- «(?'x')» добавляет в коллекцию с именем «x» один элемент;
- «(?'-x')» убирает из коллекции «x» один элемент;
- «(?(x)(?!))» проверяет, что в коллекции «x» не осталось элементов.

Язык *L*, описывающий вложенные операторы языка Pascal «**begin end**;»:

```
@"^\s*((?'begin'begin\s+)+(?'-begin'end\s*;\s*)+)*(?(begin)(?!))$".
```