Лекция 04.
Регулярные выражения

Регулярные множества

Регулярные множества – это следующие множества цепочек символов из некоторого алфавита Σ:

- Пустое множество $\emptyset$.
- Множество из пустой цепочки $\{\lambda\}$.
- Множество из любого символа $\{a\}$ алфавита Σ.
- Множество всех возможных цепочек вида $\alpha\beta$ (конкатенация); $\alpha \in A$, $\beta \in B$, где A, B – регулярные множества.
- Объединение множеств $A \cup B$, где A, B – регулярные множества.
- Объединение множеств всех возможных цепочек вида λ, α_1, $\alpha_1\alpha_2$, $\alpha_1\alpha_2\alpha_3$, . . . и т.д., где все α_1, α_2, α_3, . . . принадлежат регулярному множеству A (A^* – транзитивное замыкание A).

Регулярные выражения

Регулярное выражение – это шаблон для задания регулярного множества цепочек символов из некоторого алфавита Σ.

Кроме символов алфавита Σ в регулярное выражение могут входить вспомогательные метасимволы: Ø (пустое множество), λ (пустая строка), скобки { }, скобки { }*, вертикальная черта |.

- Пустое множество обозначается знаком Ø.
- Пустая цепочка обозначается знаком λ.
- Символ алфавита Σ обозначает себя сам.
- Если α и β – оба регулярные выражения, то запись вида αβ – регулярное выражение, обозначающее конкатенацию цепочек из α и β ;
- Если α и β – оба регулярные выражения, то запись вида α | β – регулярное выражение, обозначающее объединение множеств цепочек из α и β ;
- Если α – регулярное выражения, то запись вида {α} – то же самое регулярное выражение, рассматриваемое, как единое целое;
- Если α – регулярное выражения, то запись вида {α}* – регулярное выражение, обозначающее объединение множеств цепочек из:

λ, α, αα, ααα, . . .

и т.д., ({α}* – транзитивное замыкание α).

Примеры

1. Регулярное выражение:

{λ|+|–}{0|1|2|3|4|5|6|7|8|9}{0|1|2|3|4|5|6|7|8|9}*

задает запись целого числа без знака или со знаком «+» или «–».

Для краткости вместо явного перечисления цифр или букв через символ «|», будем использовать многоточие.

---

2. Регулярное выражение:

{0|1|. . . |9}{0|1|. . . |9}* .{0|1|. . . |9}{0|1|. . . |9}*

задает запись беззнакового десятичного числа с дробной частью.

---

3. Регулярное выражение:

{A|B|. . .|Z}{0|1|. . . |9|A|B|. . .|Z}*

задает запись идентификатора (имени) для большинства языков программирования.

Преобразование регулярного выражения к праволинейной грамматике

1. Пусть задано регулярное выражение α. Запишем прообраз порождающего правила в виде: $S \rightarrow \alpha$, где S – начальный нетерминал.
2. Пусть прообраз порождающего правила имеет вид: $A \rightarrow a\beta$,

   где A – некоторый нетерминал, a – терминал. Заменим это правило на следующие: $A \rightarrow aB$, $B \rightarrow \beta$, где B – новый нетерминал.
3. Пусть прообраз порождающего правила имеет вид:

   $A \rightarrow \{\alpha_1|\alpha_2\}\beta$, где A – некоторый нетерминал. Заменим это правило на следующие: $A \rightarrow \alpha_1\beta$, $A \rightarrow \alpha_2\beta$.
4. Пусть прообраз порождающего правила имеет вид:

   $A \rightarrow \{\alpha\}^*\beta$, где A – некоторый нетерминал. Заменим это правило на следующие: $A \rightarrow \beta$, $A \rightarrow \alpha A$.

Преобразования проводятся до тех пор, пока все полученные порождающие правила не станут праволинейными.

Реализация регулярных выражений

- Наибольшее развитие регулярные выражения получили в Perl, где их поддержка встроена непосредственно в интерпретатор.
- В VBScript и JScript используется объект RegExp, в C/C++ можно использовать библиотеки Regex++ и PCRE (Perl Compatible Regular Expression).
- Для Java существует целый набор расширений – ORO , RegExp, Rex и gnu.regexp.
- Microsoft Visual Studio.Net

Опции

/i Поиск без учета регистра.

/m Многострочный режим, позволяющий находить совпадения в начале или конце строки, а не всего текста.

/n Находит только явно именованные или нумерованные группы в форме (?<name>…). Значение этого будет объяснено ниже, при обсуждении роли скобок в регулярных выражениях.

/c Компилирует. Генерирует промежуточный MSIL-код, перед исполнением превращающийся в машинный код.

/s Позволяет интерпретировать конец строки как обыкновенный символ-разделитель. Часто это значительно упрощает жизнь.

/x Исключает из образца неприкрытые незначащие символы (пробелы, табуляция и т.д.) и включает комментарии в стиле Perl (#). Есть некоторая вероятность, что к выходу в свет эти комментарии могут исчезнуть.

/r Ищет справа налево.

Метасимволы

\ - считать следующий метасимвол как обычный символ.

^ - начало строки

. - один произвольный символ. Кроме '\n' - конец строки.

$ - конец строки

| - альтернатива (или)

() - группировка

[] - класс символов

Метасимволы

\w Слово. То же, что и [a-zA-Z_0-9].

\W Все, кроме слов. То же, что и [^a-zA-Z_0-9].

\s Любое пустое место. То же, что и [\f\n\r\t\v].

\S Любое непустое место. То же, что и [^ \f\n\r\t\v].

\d Десятичная цифра. То же, что и [0-9].

\D Не цифра. То же, что и [^0-9].

Метасимволы для последовательностей

\w+ - слово

\d+ - целое число

[+-]?\d+ - целое со знаком

[+-]?\d+\.?\d* - число с точкой

Мнимые метасимволы

\b - граница слова
\B - не граница слова
\A - начало строки
\Z - конец строки
\G - конец действия m//g

Квантификаторы

* Соответствует 0 или более вхождений предшествующего выражения. Например, 'zo*' соответствует "z" и "zoo".

+ Соответствует 1 или более предшествующих выражений. Например, "zo+" соответствует "zo" and "zoo", но не "z".

? Соответствует 0 или 1 предшествующих выражений. Например, 'do(es)?' соответствует "do" в "do" or "does".

{n} n – неотрицательное целое. Соответствует точному количеству вхождений. Например, 'o{2}' не найдет "o" в "Bob",но найдет два "o"' в "food".

{n,} n – неотрицательное целое. Соответствует вхождению, повторенному не менее n раз. Например, 'o{2,}' не находит "o" в "Bob", зато находит все "o" в "fooooood". 'o{1,}' эквивалентно 'o+'. 'o{0,}' эквивалентно 'o*'.

{n,m} m и n – неотрицательные целые числа, где n <= m. Соответствует минимум n и максимум m вхождений. Например, 'o{1,3} находит три первые "o" в "fooooood". 'o{0,1}' эквивалентно 'o?'. Пробел между запятой и цифрами недопустим.

«Жадность»

- Важной особенностью квантификаторов '*' и '+' является их всеядность. Они находят все, что смогут – вместо того, что нужно.

- Излечить квантификатор от жадности можно, добавив '?'.

Умерить аппетит

- *? - станет 0 и более
- +?- 1 и более
- ?? - 0 или 1 раз
- {n}? - точно n раз
- {n,}? - не меньше n раз
- {n,m}?- больше или равно n и меньше m раз

Дополнительные переменные

$1, $2, …

$+ - обозначает последнее совпадение

$& - все совпадение

$` - все до совпадения

$' - все после совпадения

Правила регулярного выражения

- Любой символ обозначает себя самого, если это не метасимвол. Если вам нужно отменить действие метасимвола, то поставьте перед ним '\'.
- Строка символов обозначает строку этих символов.
- Множество возможных символов (класс) заключается в квадратные скобки '[]', это значит, что в данном месте может стоять один из указанных в скобках символов. Если первый символ в скобках это '^' - значит ни один из указанных символов не может стоять в данном месте выражения. Внутри класса можно употреблять символ '-', обозначающий диапазон символов. Например, a-z - один из малых букв латинского алфавита, 0-9 - цифра и т.д.
- Все символы, включая специальные, можно обозначать с помощью '\' как в языке C.
- Альтернативные последовательности разделяются символом '|' Заметьте что внутри квадратных скобок это обычный символ.
- Внутри регулярного выражения можно указывать "подшаблоны" заключая их в круглые скобки и ссылаться на них как '\номер' Первая скобка обозначается как '\1'.

Например,

- $str=~/perl/; проверяет, есть ли в строке $str подстрока "perl"
- $str=~/^perl/; проверяет, начинается ли строка с подстроки "perl"
- $str=~/perl$/; проверяет, заканчивается ли строка на подстроку "perl"
- $str=~/c|g|i/; проверяет, содержит ли строка символ 'c' или 'g' или 'i'
- $str=~/cg{2,4}i/;проверяет, содержит ли строка символ 'c', следующие сразу за ним 2-4 символа 'g', за которыми следует символ 'i'

Следующая тема:

«Конечные автоматы»