A crash course on regular expressions and regular grammars

Professor Hossein Saiedian

EECS 348: Software Engineering

Spring 2023

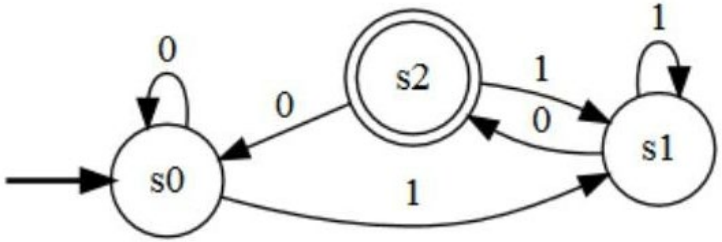
THE UNIVERSITY OF KANSAS

FSM: Basic components

- A set of states, S
  - An initial state S_0
  - A possible set of one or more final states (F)
  - The initial and final states can be the same
- A set of inputs, I, also called the input vocabulary
- A set of outputs, O, also called the output vocabulary
- A finite-state automaton *accepts* a string *x* if $\delta(S_0, x) \in F$; i.e., the finite state automaton ends up in a final state

FSM: Basic components

- A finite state automaton (FSA) is a graph with directed labeled arcs, two types of nodes, final and a unique start state
  - An FSA can have more than one final state
  - Also called a state machine
  - Deterministic FSA: For each state and for each input alphabet, there is exactly one transition
- The language accepted by a FSM is set of strings that move from start node to a final node



A couple of related terms

- **BNF** (Backus-Naur Form): a formal meta language used in defining context-free grammars, e.g., in compiler design
  - <terminal>
  - Defined as: ::=
  - Alternative: |
  - Optional: [ ]
  - Combining: ( )
  - Maybe: ?
  - Repeated: (…)* some also suggest {…}
  - One or more: (…)+
  - Recursion: <integer> ::= <digit> | <integer><digit>

Some symbols are just for convenience

Some BNF examples

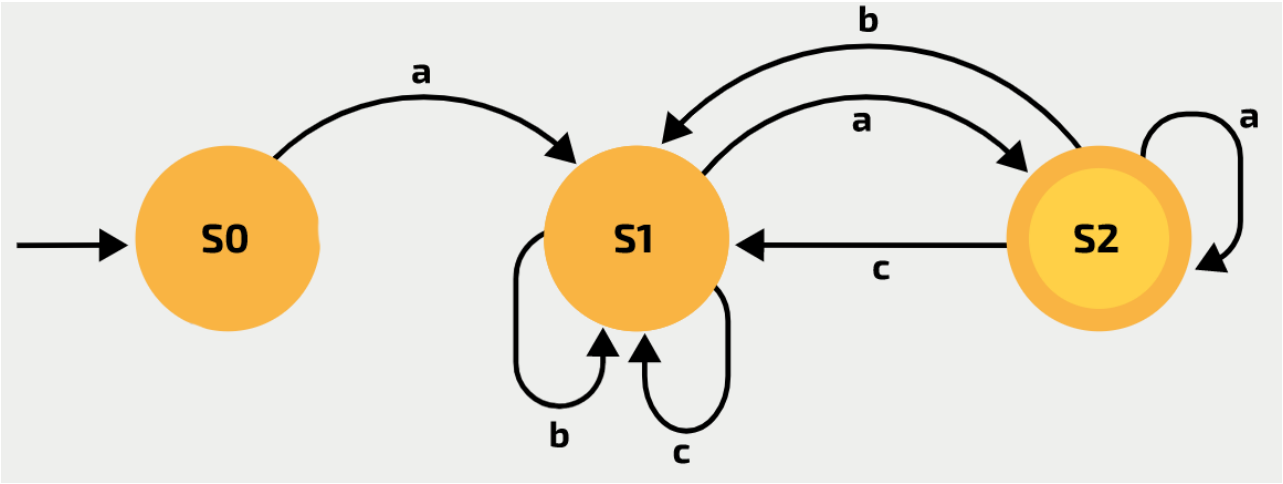
```
NUMBER ::=  WHOLE_PART  FRACTION_PART?
WHOLE_PART ::=  NUMERAL_SEQUENCE
FRACTION_PART ::= “.” NUMERAL_SEQUENCE
NUMERAL_SEQUENCE ::=  NUMERAL  NUMERAL*

NUMERAL ::=  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
EXPRESSION ::= NUMERAL | (EXPRESSION OPTR EXPRESSION)
OPTR ::=  +  |  -
```

Regular expressions

- Not as powerful as BNF (e.g., no recursion)
- Regular expressions can express finite languages by defining a pattern for finite strings of symbols
- The grammar defined by regular expressions is known as regular grammar
- A regular expression is a pattern that can be recognized by a FSM
  - Regular expressions and FSMs are equivalent concepts
  - Semantically the same; one is textual (metal language), one is graphical

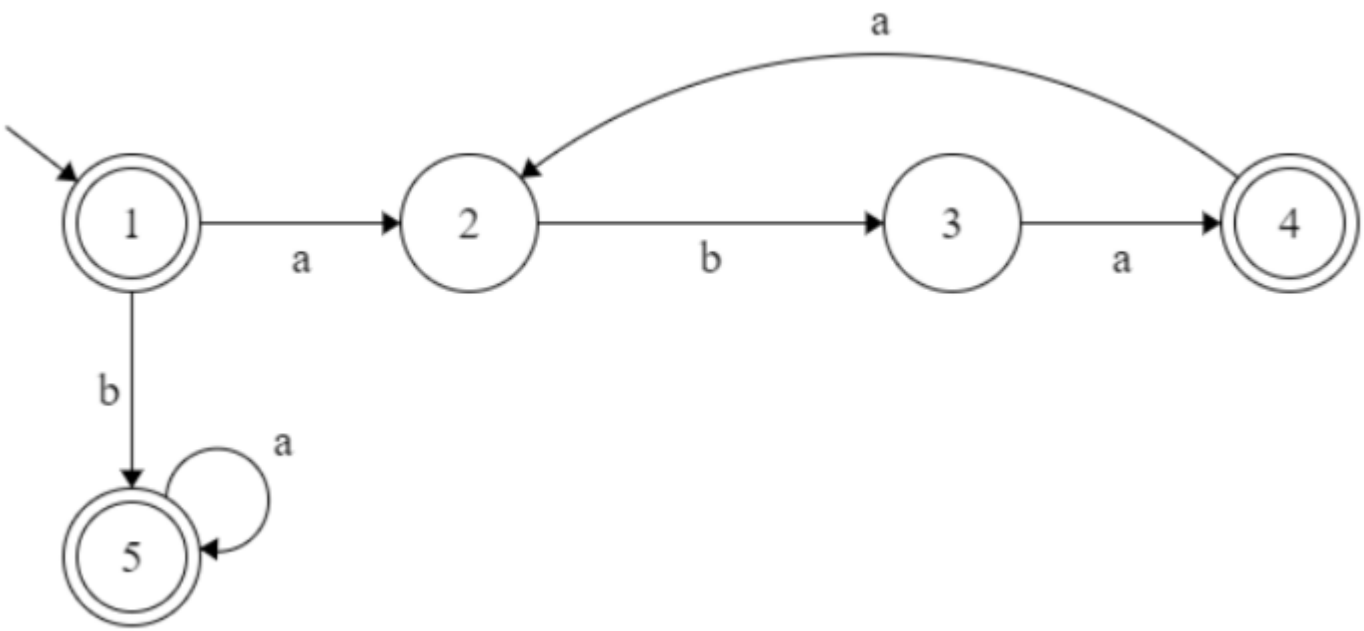
Examples of regular expressions

```
a(a|b|c)*a
```



Examples of regular expressions

- $(aba)^+|ba^*$



Examples of regular expressions

- What do these represent?

L ::= “A” | “B” | … | “a”| “b”| …

D ::= 1 | 2 | …

L(“_”|L|D)*

(“+”|“-”)DD*

(“+”|“-”)D+ “.” D+ [(“E”|“e”)[“+”|“-”] D+]

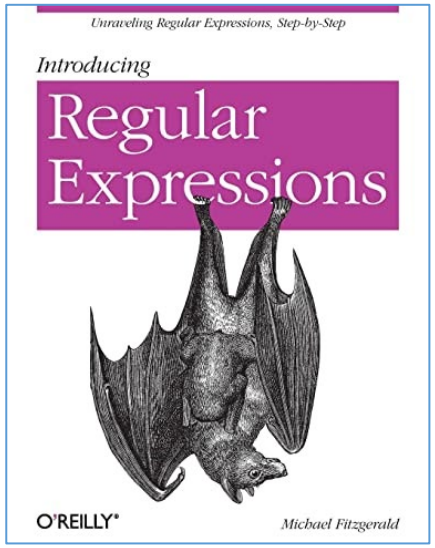
Regular expression in Unix

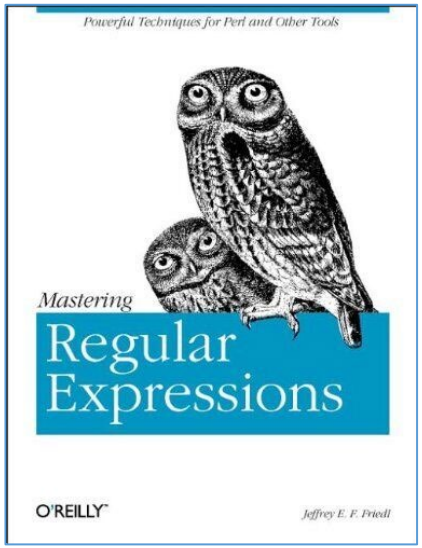
- A very similar concept, slightly different notation
- Extremely powerful for pattern matching (vim, grep, sed, …)
- Common Unix symbols

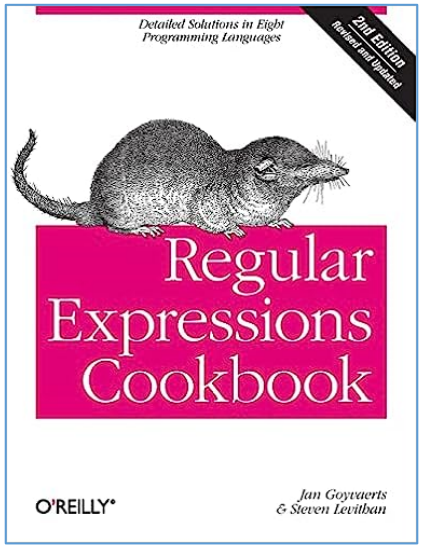
Symbol	Descriptions
.	replaces any character
^	matches start of string
$	matches end of string
*	matches up zero or more times the preceding character
\	Represent special characters
()	Groups regular expressions
?	Matches up exactly one character

Regular expressions in Unix

- A very similar concept, slightly different notation
- Extremely powerful for pattern matching (vim, grep, sed, …)

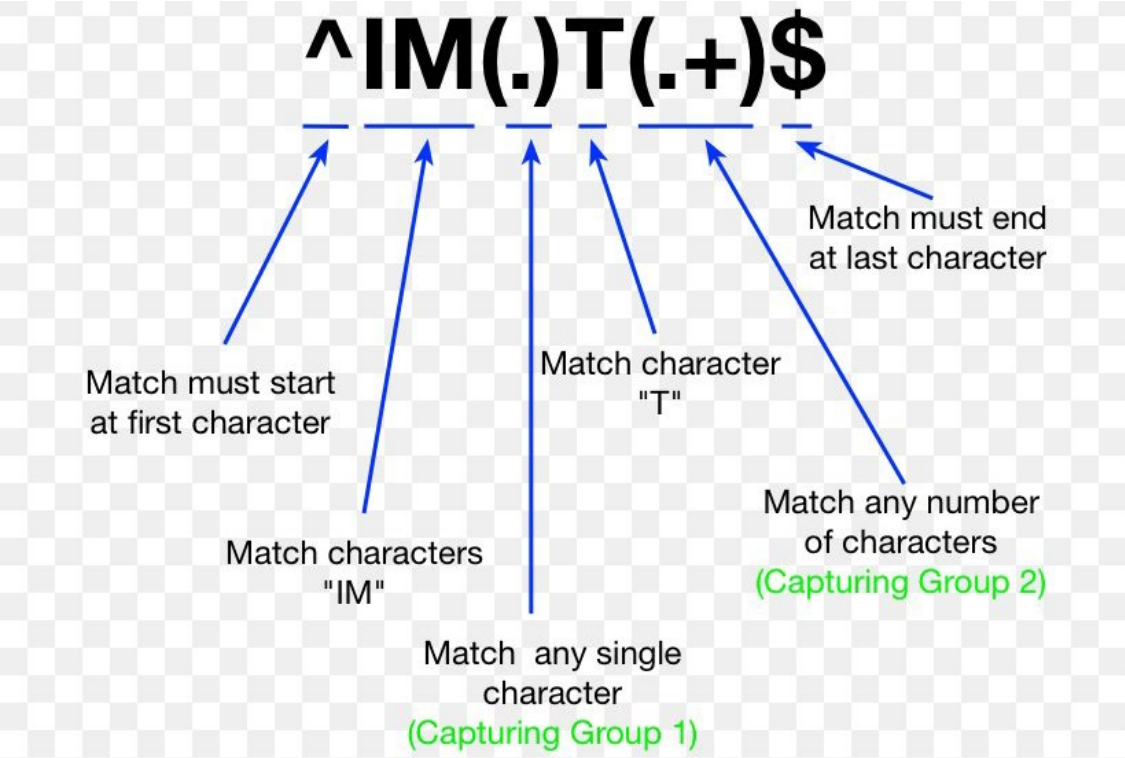






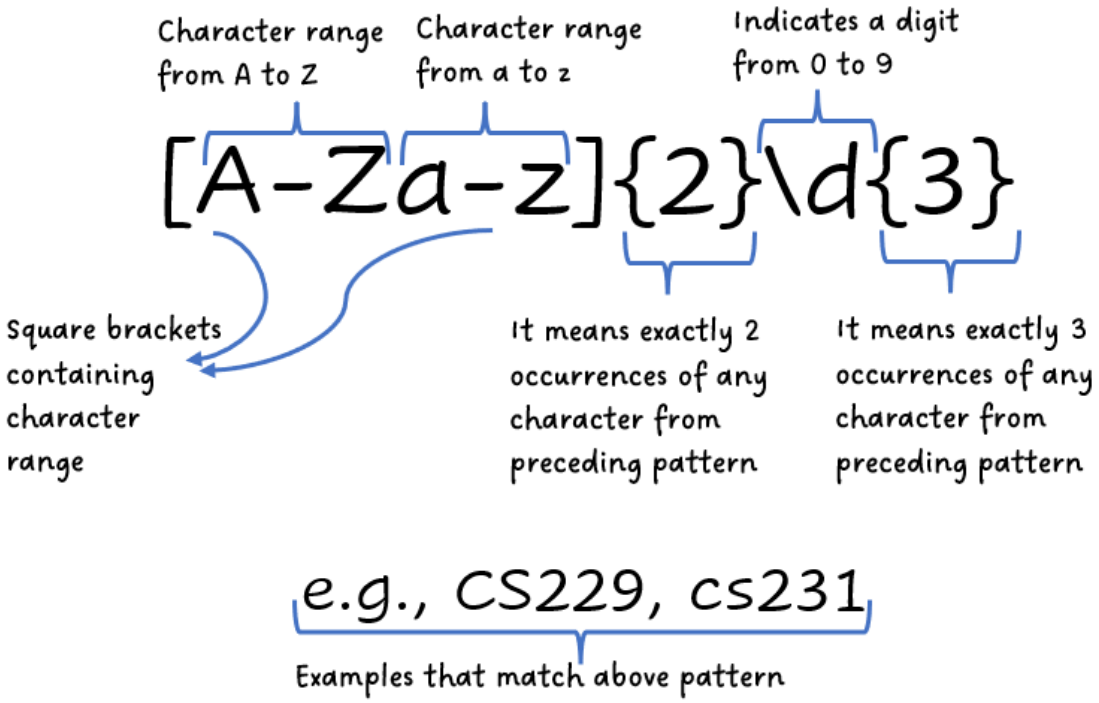
Regular expressions in Unix

- A very similar concept, slightly different notation
- Extremely powerful for pattern matching (vim, grep, sed, …)



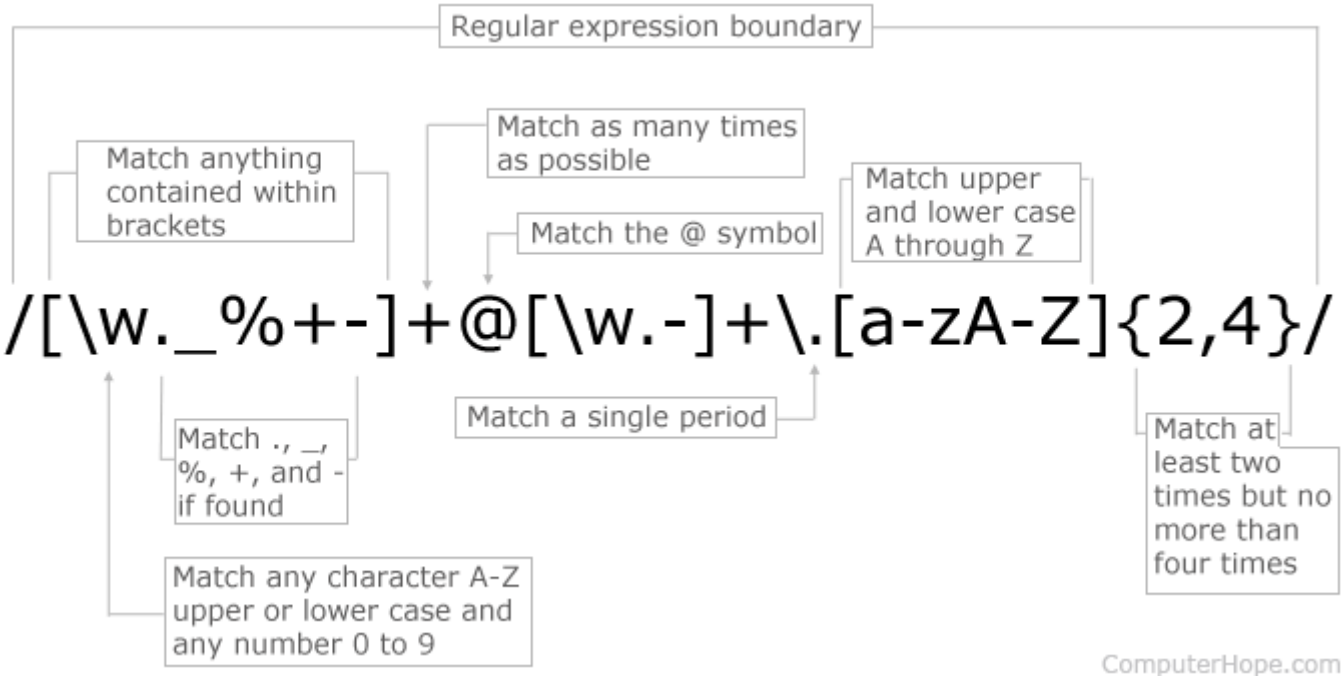
Regular expressions in Unix

- A very similar concept, slightly different notation
- Extremely powerful for pattern matching (vim, grep, sed, …)



Regular expressions in Unix

- A very similar concept, slightly different notation
- Extremely powerful for pattern matching (vim, grep, sed, …)

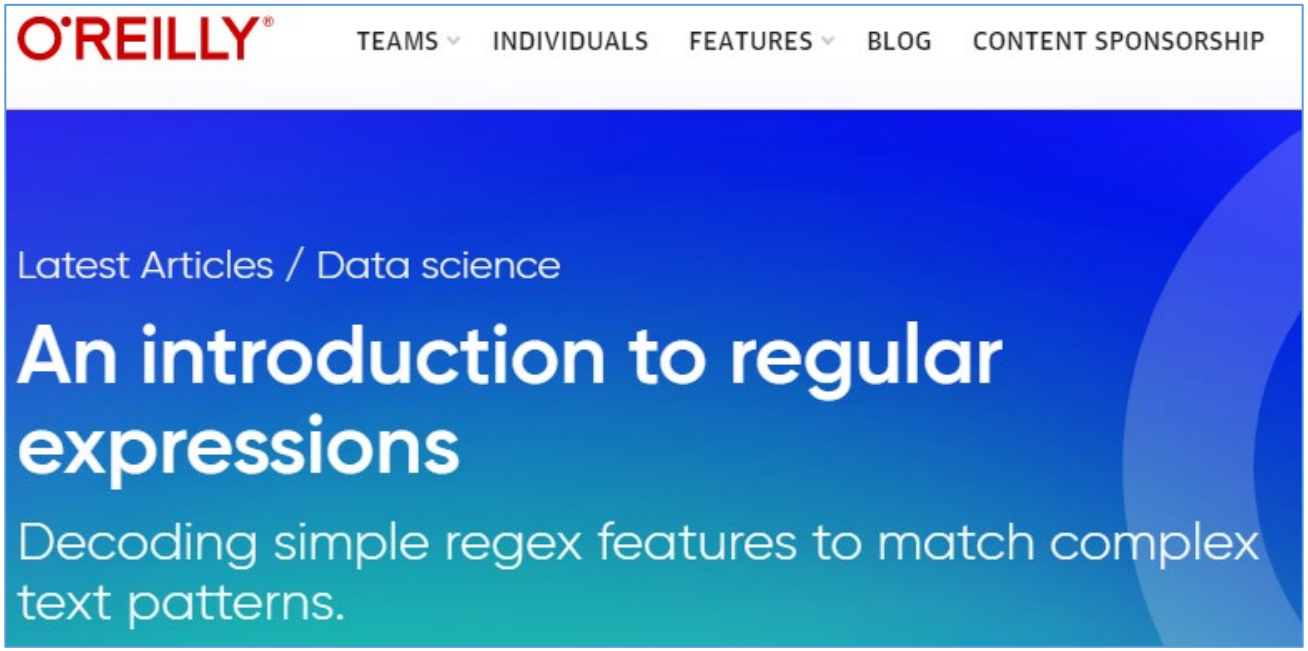


A short RE cheat sheet

[abc]	A single character: a, b or c
[^abc]	Any single character but a, b, or c
[a-z]	Any single character in the range a-z
[a-zA-Z]	Any single character in the range a-z or A-Z
^	Start of line
$	End of line
\A	Start of string
\z	End of string
.	Any single character
\s	Any whitespace character
\S	Any non-whitespace character
\d	Any digit
\D	Any non-digit
\w	Any word character (letter, number, underscore)
\W	Any non-word character
\b	Any word boundary character
(...)	Capture everything enclosed
(a|b)	a or b
a?	Zero or one of a
a*	Zero or more of a
a+	One or more of a
a{3}	Exactly 3 of a
a{3,}	3 or more of a
a{3,6}	Between 3 and 6 of a

Regular expressions in Unix

- A very similar concept, slightly different notation
- Extremely powerful for pattern matching (vim, grep, sed, …)



https://www.oreilly.com/content/an-introduction-to-regular-expressions