

Regular Expressions

,

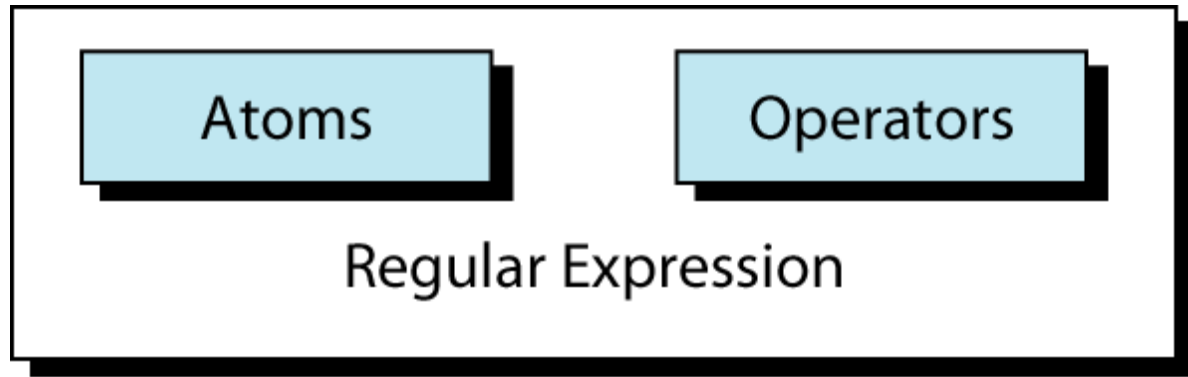
Topics

- What is a regular expression
- Regular expression components
- Construct regular expressions

Define Regular Expressions

- A pattern of special characters used to match strings in a search
- Typically made up from special characters called metacharacters
- Regular expressions are used throughout Unix/Linux
 - Editors: vim, ex, ed
 - Utilities: grep, egrep, sed, and awk

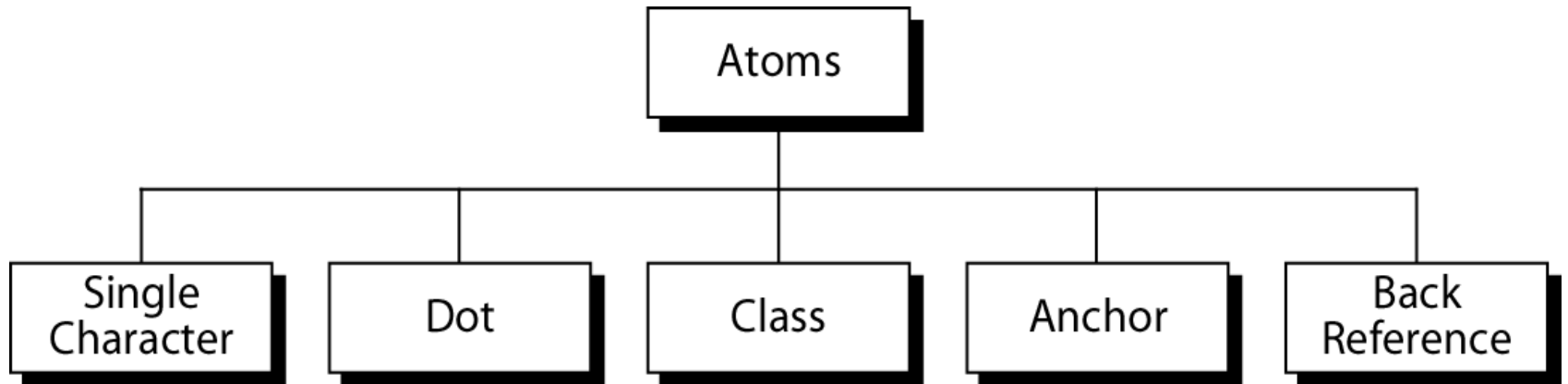
Regular Expression Forms



- An atom specifies what text is to be matched and where it is to be found
- An operator combines regular expression atoms

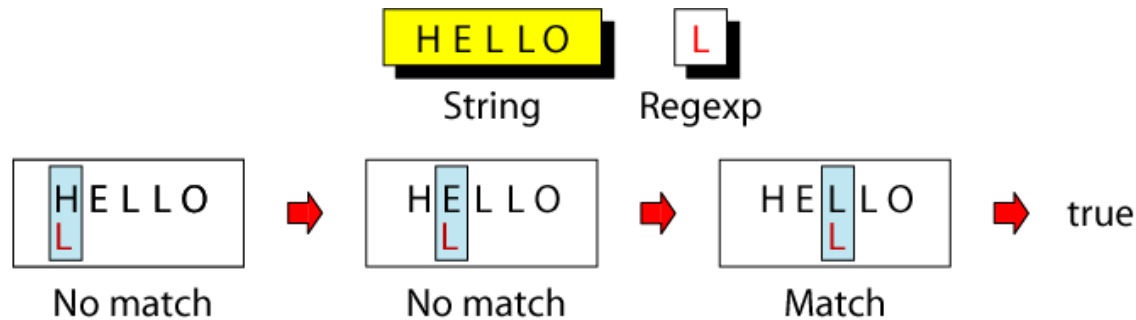
Atoms

An atom can be one of 5 types

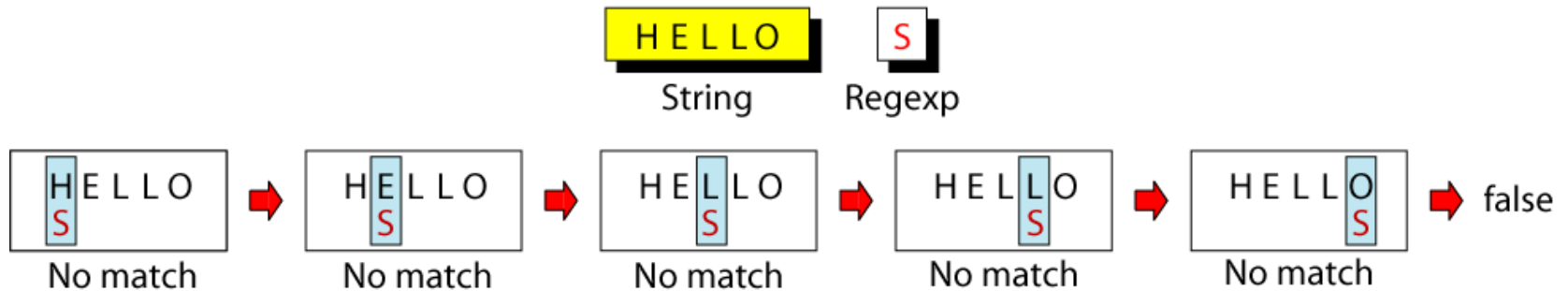


Single Character Atom

A single character matches itself



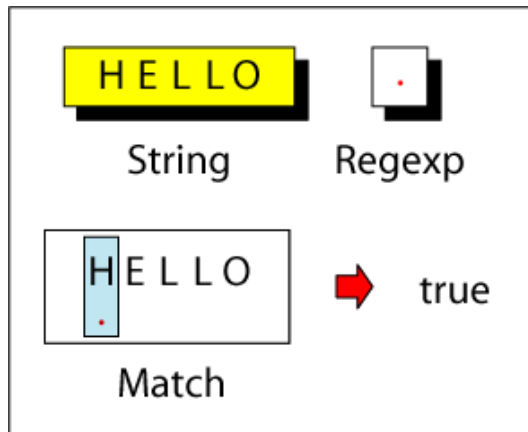
(a) Successful Pattern Match



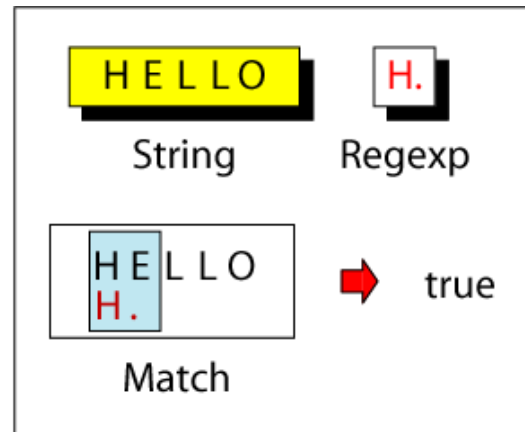
(b) Unsuccessful Pattern Match

Dot Atom

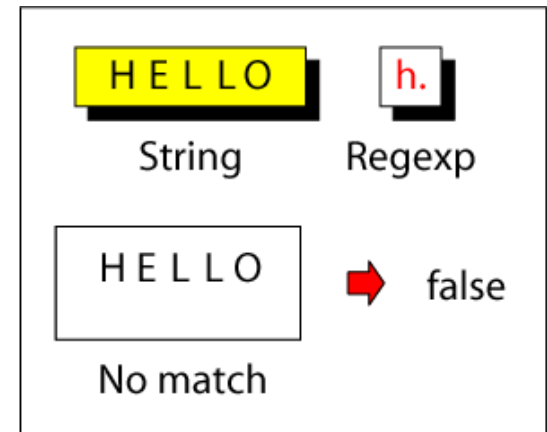
Matches any single character except the new line character



(a) Single-Character



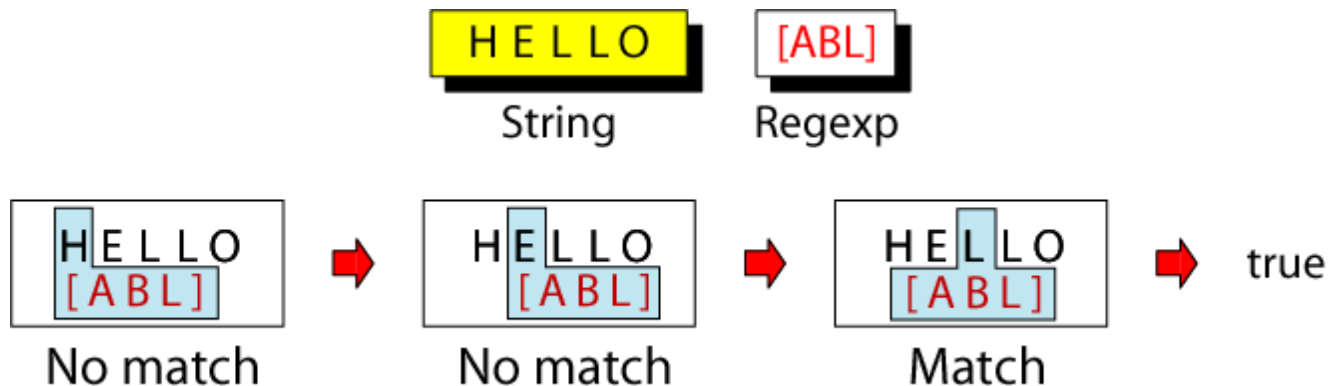
(b) Combination-True



(c) Combination-False

Class Atom

- Matches any one of the characters in a set
 - 1) A range of characters is indicated by a dash, e.g. [A-Q]
 - 2) Can specify characters to be excluded from the set, e.g. [^0-9] matches any character other than a number.



Class Examples

RegExpr

[A-H]



[ABCDEFGH]

[A-Z]



Any uppercase
alphabetic

[0-9]



Any digit

[a]



[or a

[0-9\ -]



digit or hyphen

RegExpr

[^AB]



Any character
except A or B

[A-Za-z]



Any alphabetic

[^0-9]



Any character
except a digit

[]a]



] or a

[^\^]





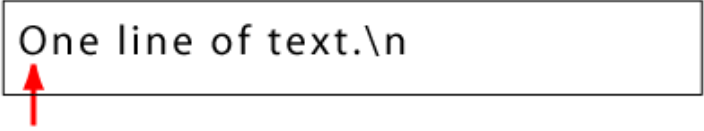


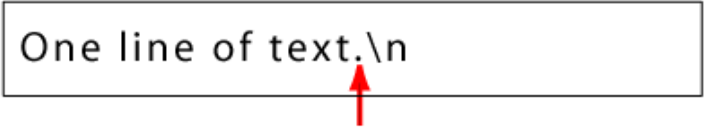





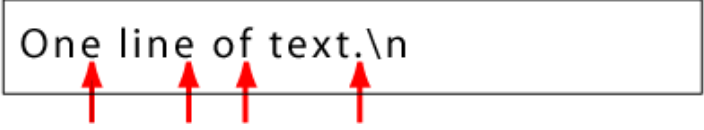
Anything except^

Short-hand Classes

- `\w` word
- `\W` non-word
- `\s` space
- `\S` non-space
- `\d` digit
- `\D` non-digit

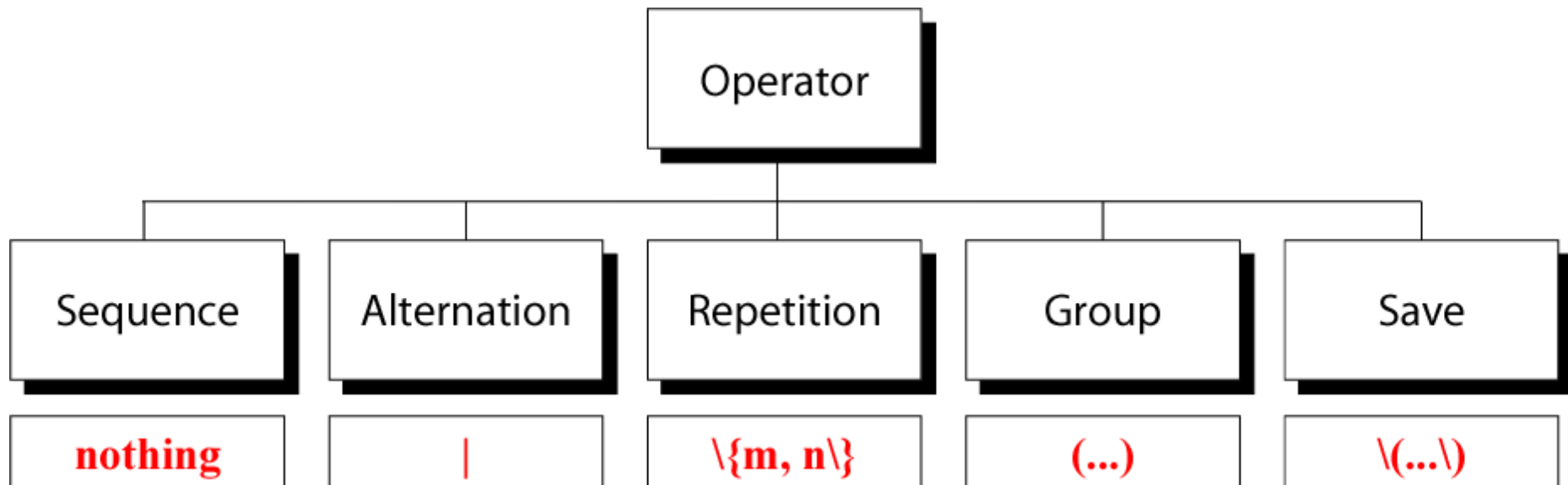
Anchors

Tell where the next character in the pattern must be located in the text data

Anchor		Means	Example
		Beginning of line	
		End of line	
		Beginning of word	
		End of word	

Operators

Link atoms

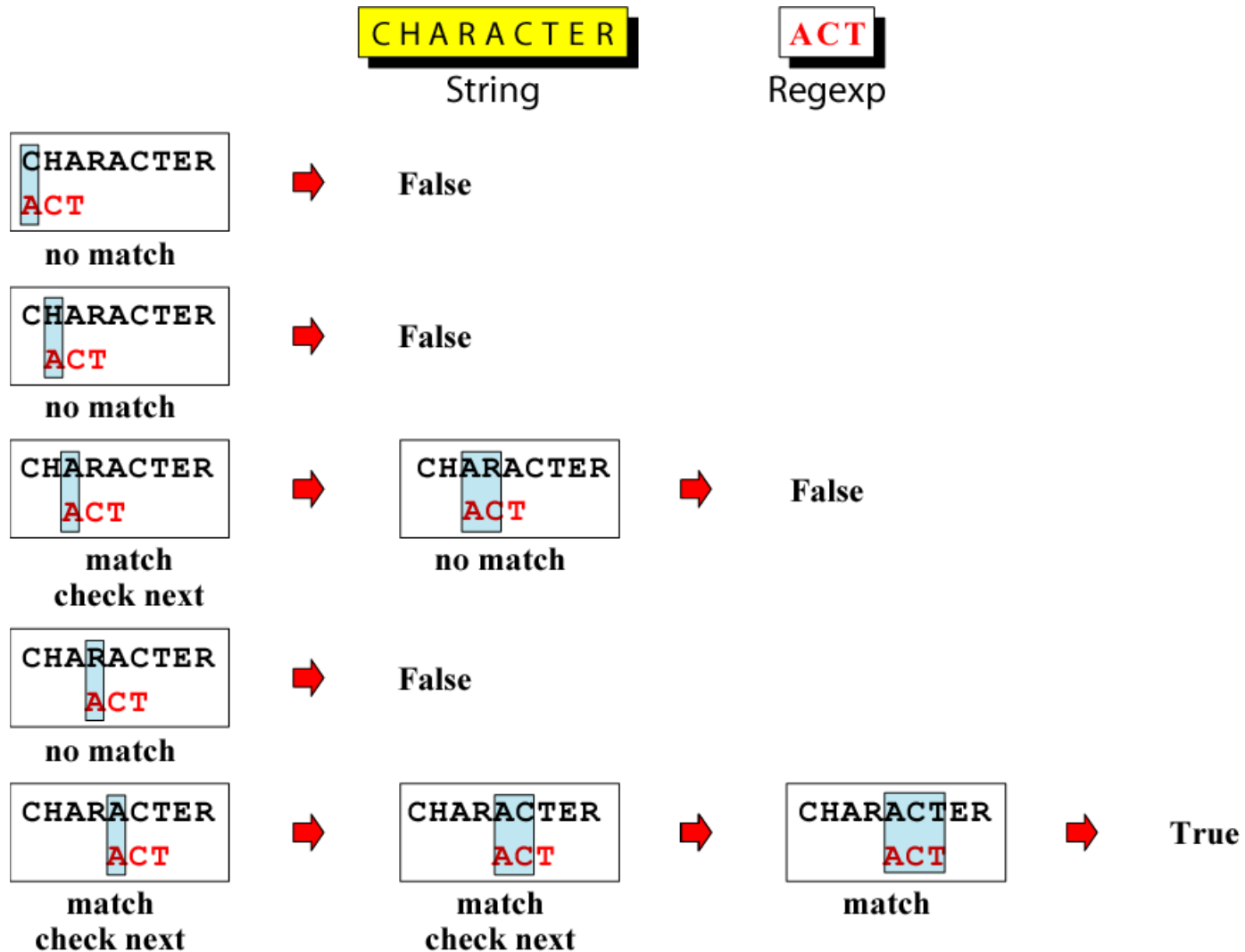


Sequence Operators

- Sequence operators are invisible
- Match all the characters from left to right sequentially

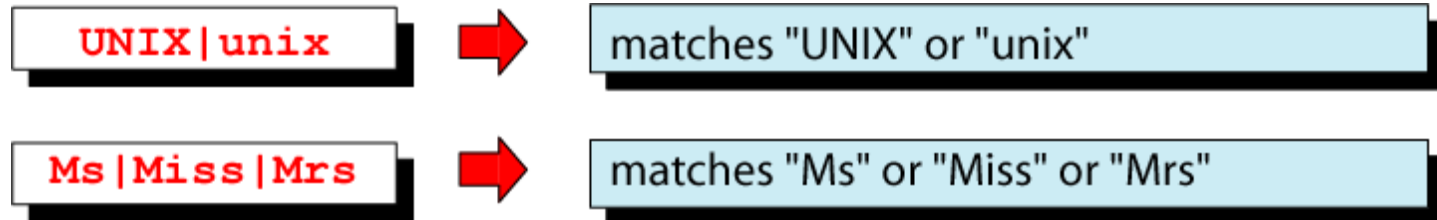
<code>dog</code>	→	matches the pattern "dog"
<code>a..b</code>	→	matches "a" , any two characters, and "b"
<code>[2-4][0-9]</code>	→	matches a number between 20 and 49
<code>[0-9][0-9]</code>	→	matches any two digits
<code>^\$</code>	→	matches a blank line
<code>^.\$</code>	→	matches a one-character line
<code>[0-9]-[0-9]</code>	→	matches two digits separated by a "-"

Sequence Operators at Work



Alternation Operator

- The alternation operator (|) is used to define one or more alternatives



Matching Alternation Operator

HELLO

String

FE | EL

Regex

H	E	L	L	O
F	E			

No match



H	E	L	L	O
	F	E		

No match



...



H	E	L	L	O
			F	E

No match



Retry

H	E	L	L	O
E	L			

No match



H	E	L	L	O
	E	L		

Match



True

Repetition Operators

- Specify how many times the immediate proceeding character may be repeated

`\{m , n\}`

matches previous character m to n times.

`A\{3 , 5\}`



matches "AAA", "AAAA", or "AAAAA"

`BA\{3 , 5\}`



matches "BAAA", "BAAAA", or "BAAAAA"

Basic Repetition Forms

Formats

`\{m\}`



matches previous atom exactly m times

`\{m, \}`



matches previous atom m times or more

`\{, n\}`



matches previous atom n times or less

Examples

`CA\{5\}`



CAAAAA

`CA\{3, \}`



CAAA, CAAAA, CAAAAA, ...

`CA\{, 2\}`



C, CA, CAA

Short Form Repetition Operators

Formats

*



special case: matches previous atom zero or more times

+



special case: matches previous atom one or more times

?



special case: matches previous atom 0 or one time only

Examples

BA*



B, BA, BAA, BAAA, BAAAA, ...

B.*



B, BA ... BZ, BAA ... BZZ,
BAAA ... BZZZ, ...

.*



zero or more characters

.+



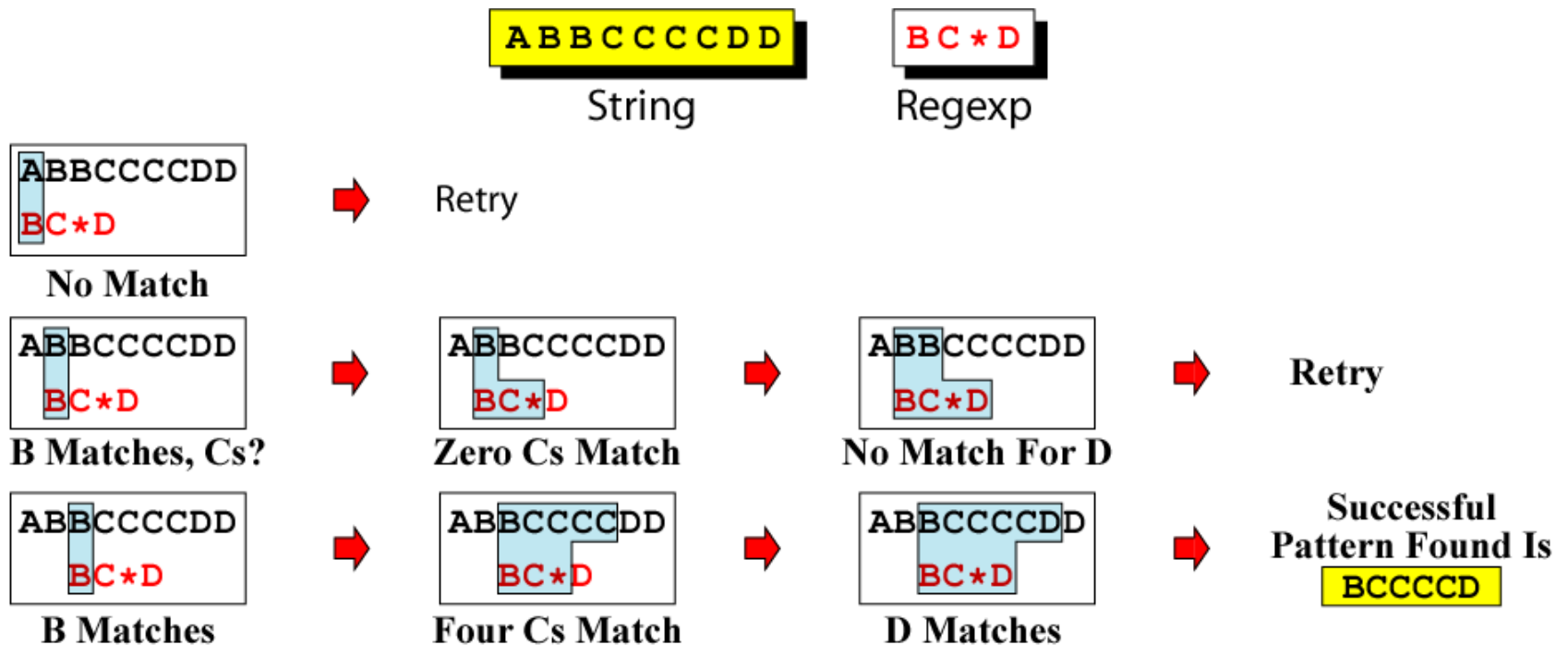
one or more characters

[0-9]?



zero or one digit

Repeating Pattern Matching



Greedy Matching

XAYXFOONFOOAX

String

A . * F O O

Regexp

(a)

XAYXFOONFOOAX
A . * F O O



No Match

(b)

XAYXFOONFOOAX
A . * F O O



Start Match

(c)

XAYXFOONFOOAX
A . * F O O



Over Greedy Match

(d)

XAYXFOONFOOAX
A . * F O O



Final Match (Greedy)

AYXFOONFOO

Matched Pattern

Stop Being Greedy

- The easiest solution to the greedy behavior of regular expression is to add a question mark

Group Operators

In the group operator, when a group of characters is enclosed in parentheses, the next operator applies to the whole group, not only the previous characters

Regex		Matches
<code>A(BC)\{3\}</code>	→	<code>ABCBCBC</code>
<code>(F(BC)\{2\}G)\{2\}</code>	→	<code>FBCBCGFBCBCG</code>

Save Operator

NACDEXGHIJABC

String

\ ([A-Z] \) . * \ 1

Regex

(a)

NACDEXGHIJABC
 \ ([A-Z] \) . * \ 1



Save N in Buffer 1

N

Buffer 1

(b)

NACDEXGHIJABC
 \ ([A-Z] \) . * \ 1



No Match for N

(c)

NACDEXGHIJABC
 \ ([A-Z] \) . * \ 1



Save A in Buffer 1

A

Buffer 1

(d)

NACDEXGHIJABC
 \ ([A-Z] \) . * \ 1



Over Greedy Match

(e)

NACDEXGHIJABC
 \ ([A-Z] \) . * \ 1



Match Text (A) in Buffer 1

ACDEXGHIJA

Matched Pattern

Saving and Grouping Metacharacters

Metacharacter	Meaning
\(. . . \)	Group subpattern and save as \1, \2, ..., \9
\n	Use saved n th subpattern
&	The whole matched pattern
~	The previous substitute string

A Few Ways to Slice and Dice a Date

	1	2	3	4	5	6	7	8	9
	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31								

「31|[123]0|[012]?[1-9]」

	1	2	3	4	5	6	7	8	9
	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31								

「0[1-9]|[12][0-9]?|3[01]?|[4-9]」

	1	2	3	4	5	6	7	8	9
	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31								

「[12][0-9]|3[01]|0?[1-9]」

Lookahead & Lookbehind

- Lookahead and lookbehind, collectively called **lookaround**, match characters before and after, then give up, returning only true or false.
 - Positive lookahead: `(?=regex)`
 - Negative lookahead: `(?!regex)`
 - Positive lookbehind: `(?<=regex)`
 - Negative lookbehind: `(?<!regex)`

Online Bash Reference

See

<http://www.gnu.org/software/bash/manual/bashref.html>

Regex Engines

- DFA (Deterministic Finite Automation)
 - The longest leftmost matches
- NFA (Non-deterministic Finite Automation)
 - Traditional NFA
 - POSIX NFA
- Hybrid DFA/NFA

Summary

- Shell provides a character-based user interface for the Unix kernel
- A shell is both a command interpreter and a programming language
- The shell also supports the following features:
 - Variable substitution
 - Filename generation
 - Pipes
 - Input/output redirection
 - User environment customization