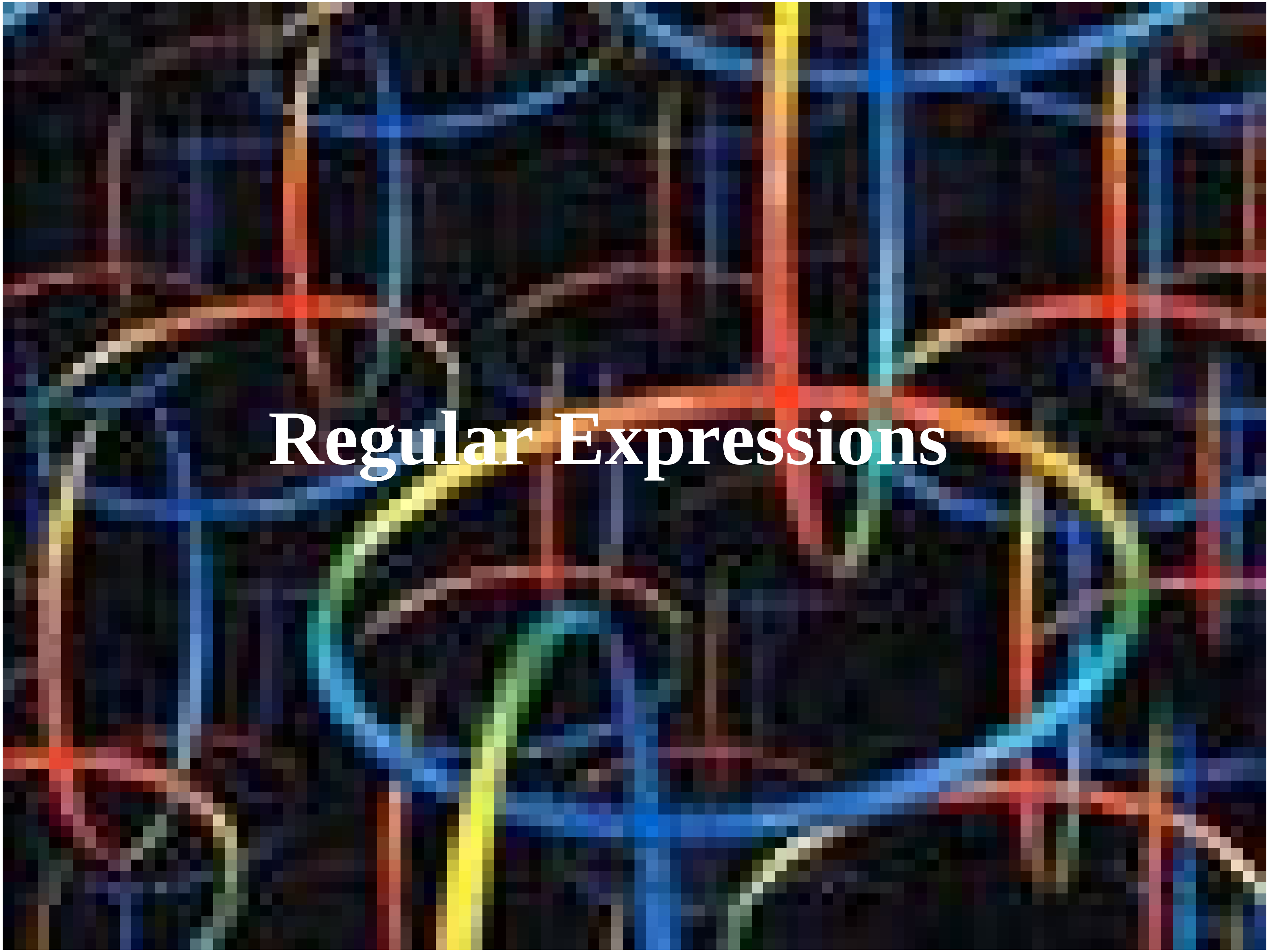


Regular Expressions

A regular expression is a pattern consisting of a sequence of characters that matched against the text.

UNIX evaluates text against the pattern to determine if the text and the pattern match.

If they match, the expression is true and a command is executed.

Some of the most powerful UNIX utilities , such as grep and sed, use regular expressions.

Regular Expression

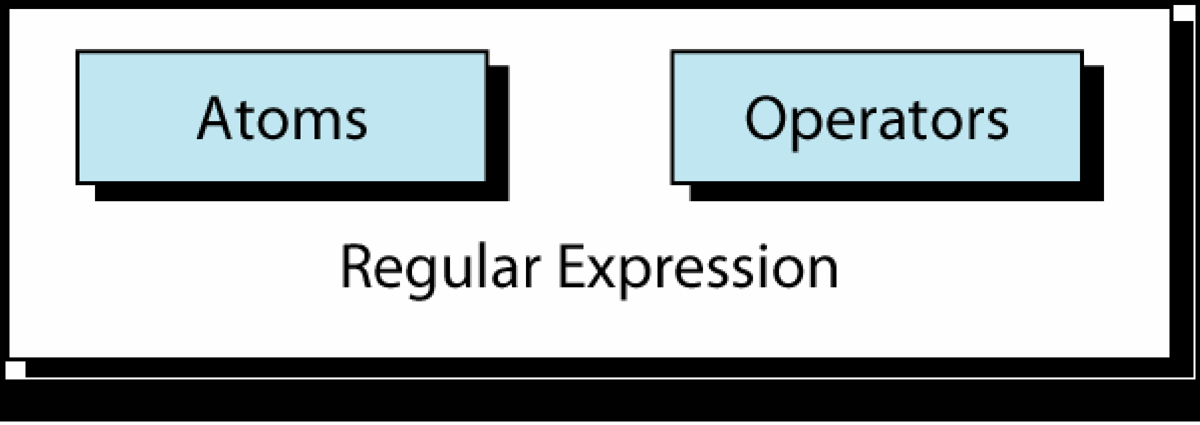
A regular expression is like a mathematical expression.

A mathematical expression is made of operands (data) and operators.

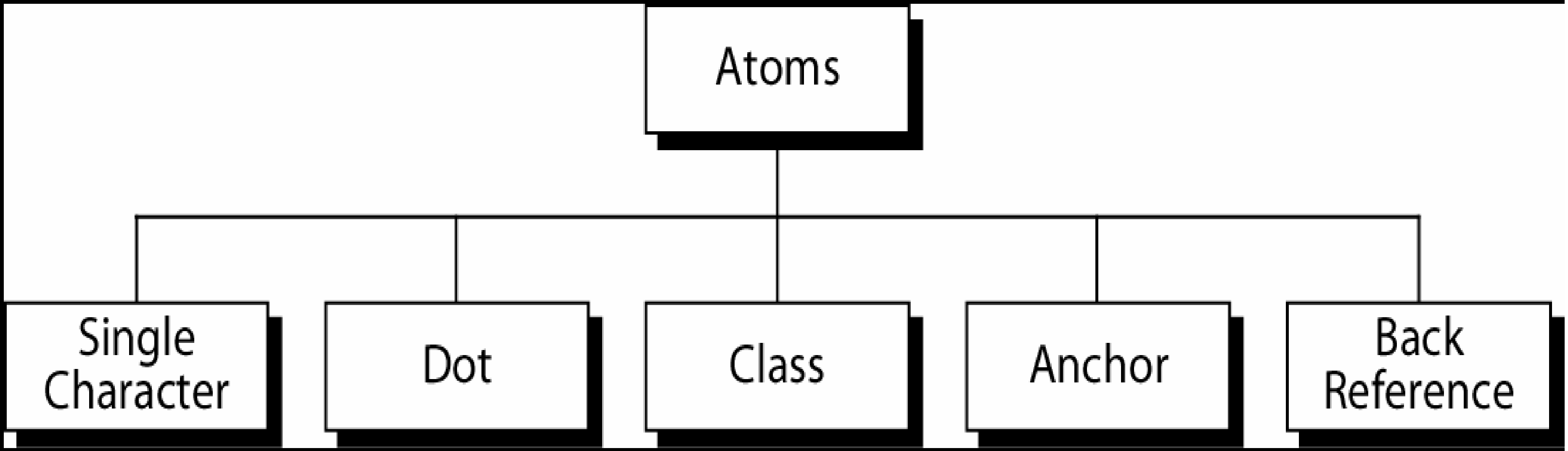
A regular expression is made of atoms and operators.

The atom specifies what we are looking for and where in the text the match is to be made.

The operator combines atoms into complex expressions.

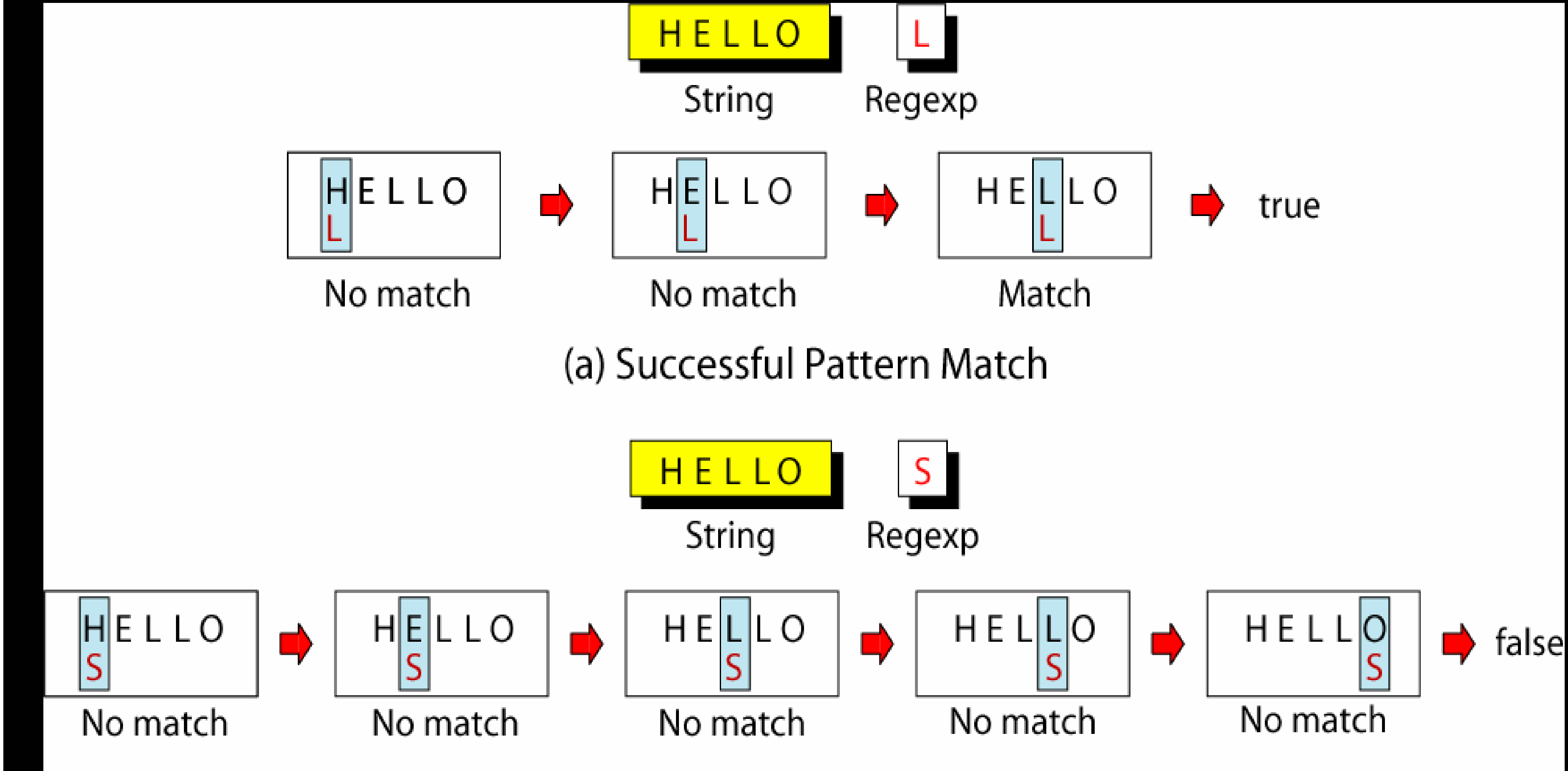


Atoms



Single-Character Pattern Example

The simplest atom is a single character.

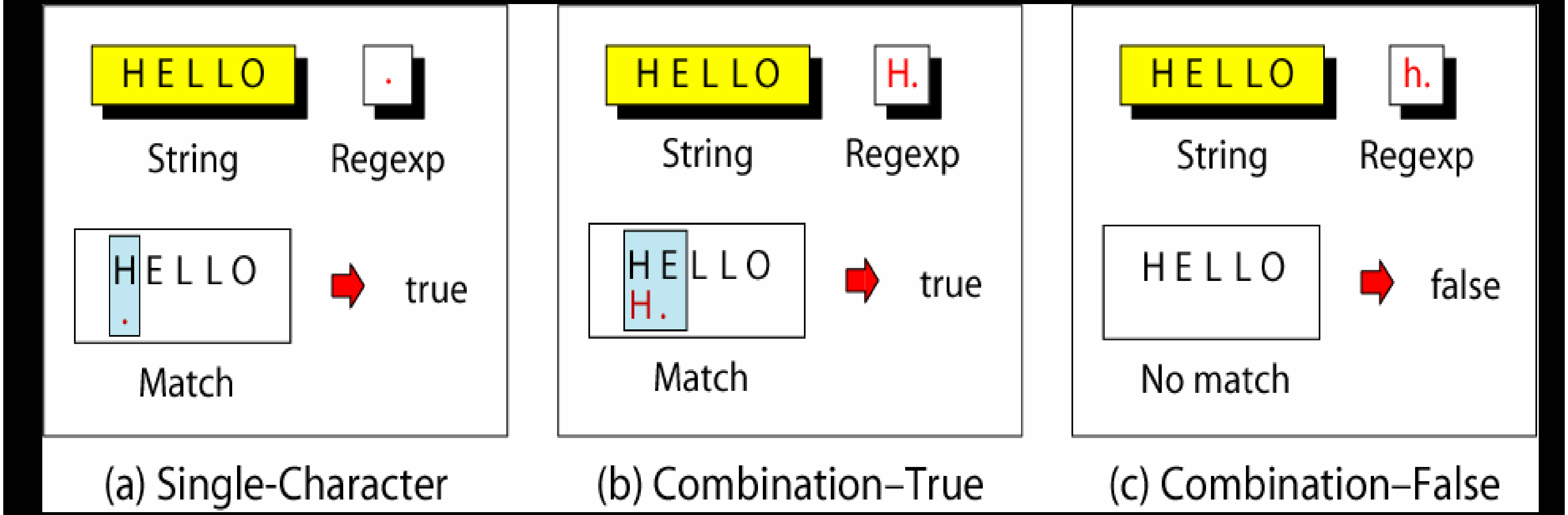


(a) Successful Pattern Match

(b) Unsuccessful Pattern Match

Dot Atom Example

A dot matches any single character except the new line character (\n).



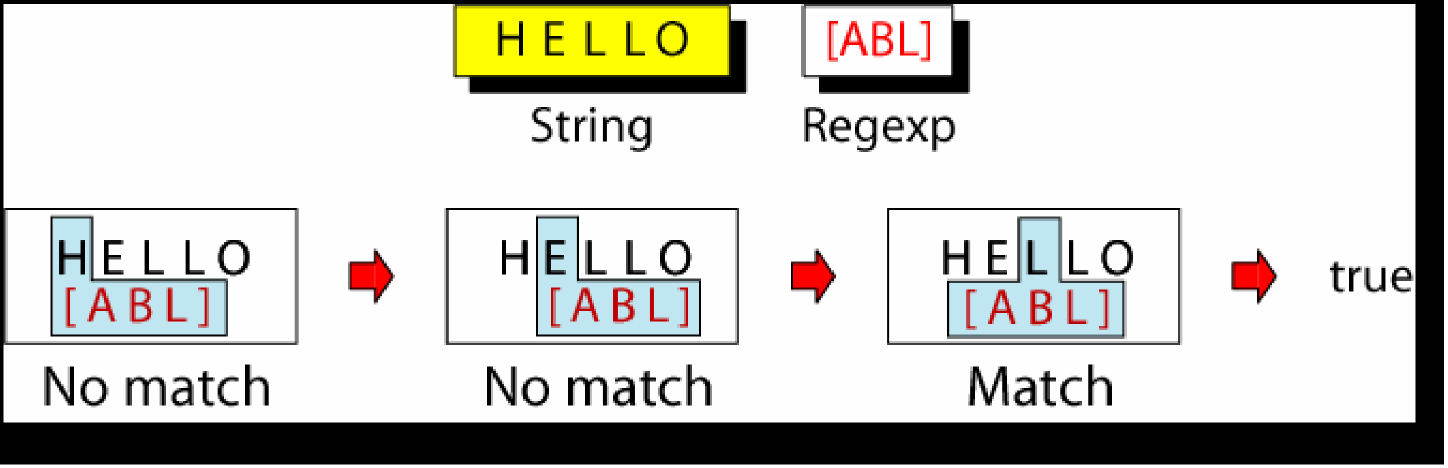
Class Atom Example

The class atom defines a set of ASCII characters, any one of which may match any of the characters in the text.

The character set to be used in the matching process is enclosed in brackets.

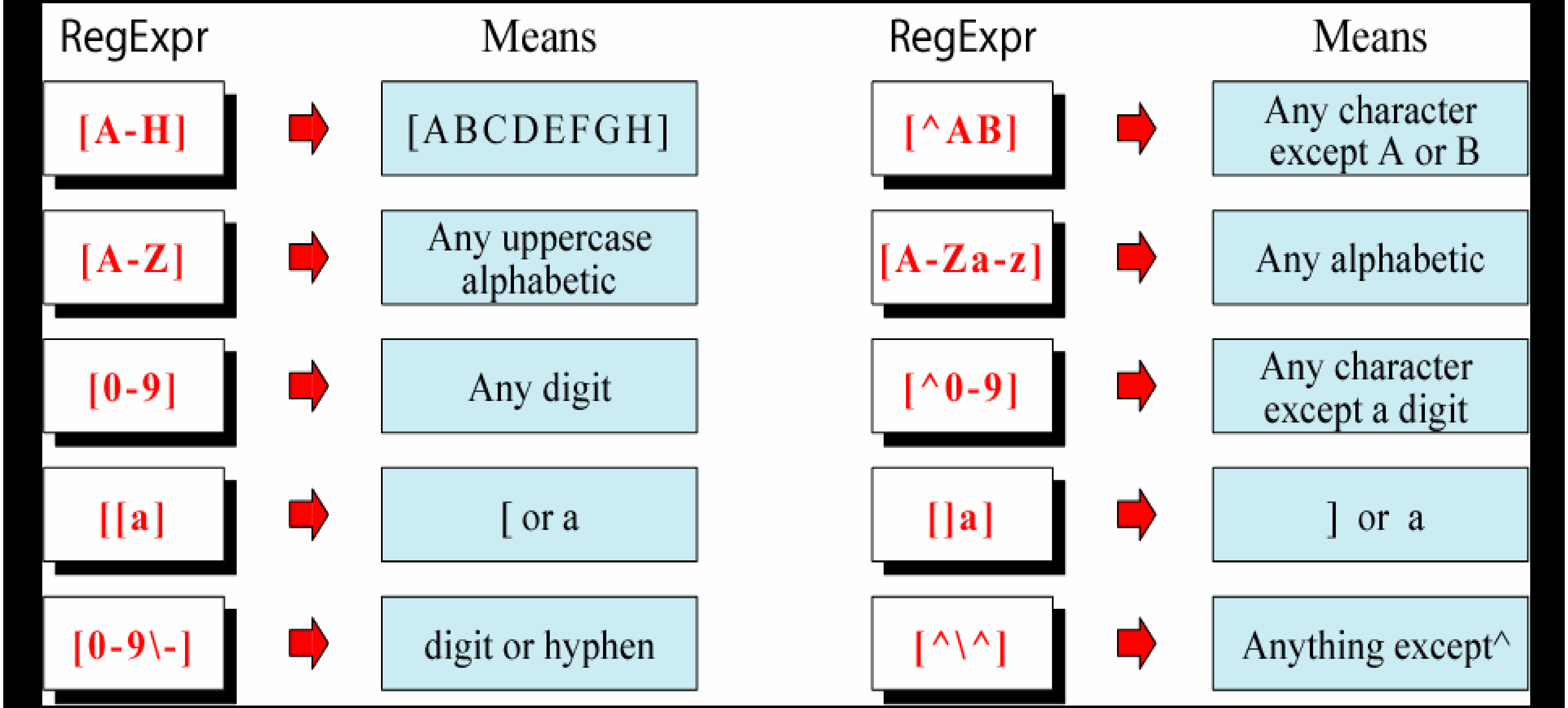
A range of text characters is indicated by a dash (-). [a-d]

^ is an exclusion operator. To specify any character other than a vowel, we use [^aeiou].



Example of Classes

The escape character (\) is used when the matching character is one of the other two tokens: – and ^.

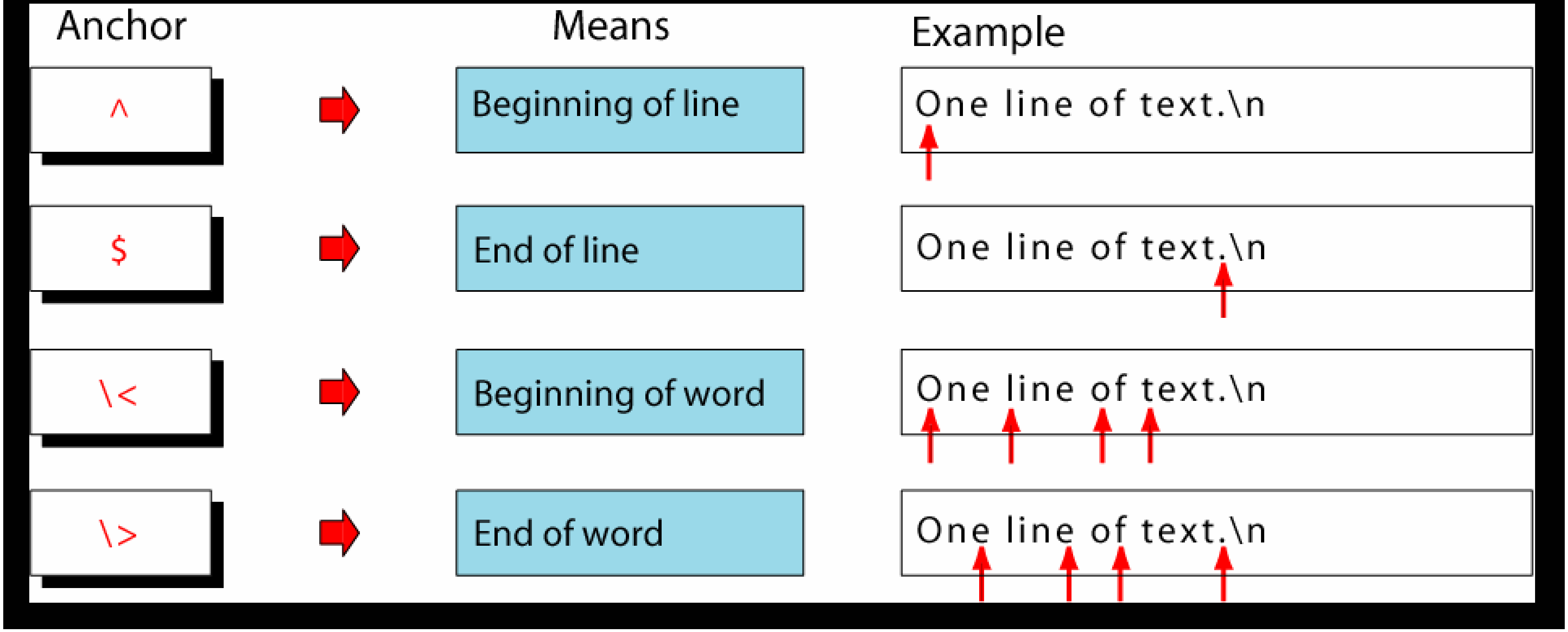


RegExpr	Means	RegExpr	Means
[A-H]	[ABCDEFGH]	[^AB]	Any character except A or B
[A-Z]	Any uppercase alphabetic	[A-Za-z]	Any alphabetic
[0-9]	Any digit	[^0-9]	Any character except a digit
[[a]	[or a	[]a]	] or a
[0-9\-]	digit or hyphen	[^\^]	Anything except^

Anchors

Anchors are atoms that are used to line up the pattern with a particular part of a string.

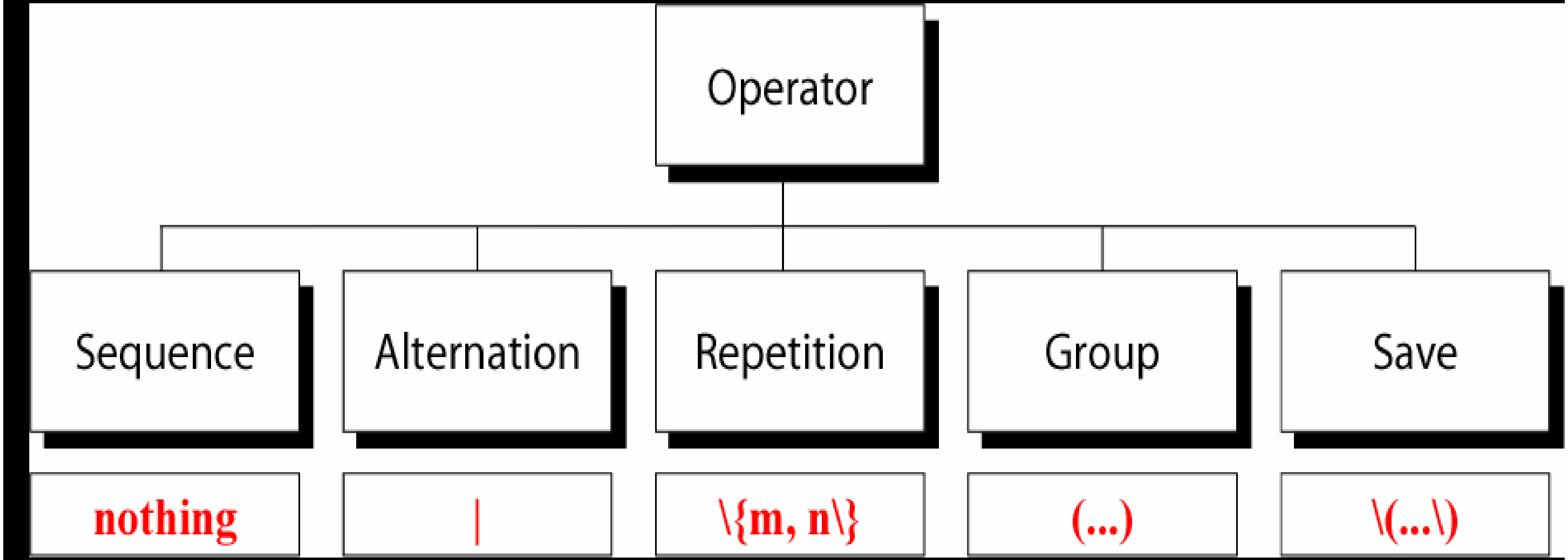
Anchors are not matched to the text, but define where the next character in the pattern must be seen.



Anchor	Means	Example
^	Beginning of line	One line of text.\n
$	End of line	One line of text.\n
\<	Beginning of word	One line of text.\n
\>	End of word	One line of text.\n

Operators

We can combine atoms with operators.



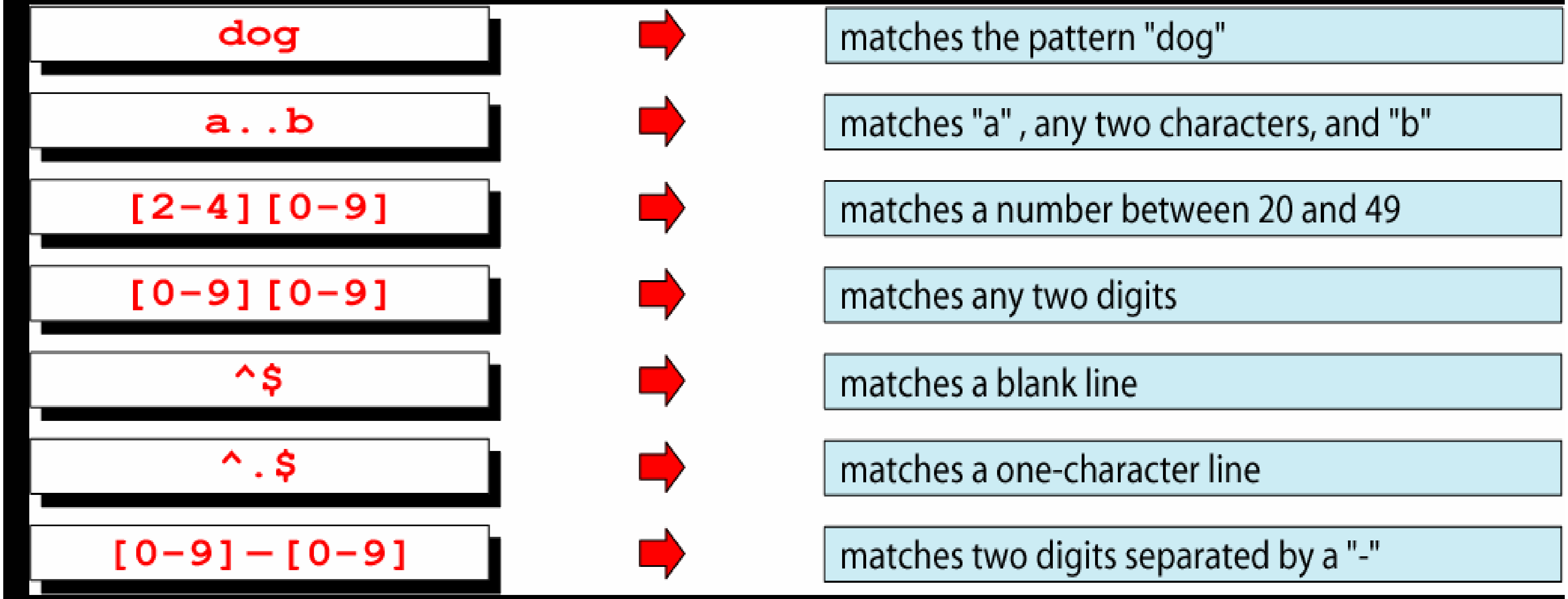
Operator

Sequence	Alternation	Repetition	Group	Save
nothing	**|**	**\{m, n\}**	**(...)**	**\(...\)**

Example of Sequence Operator

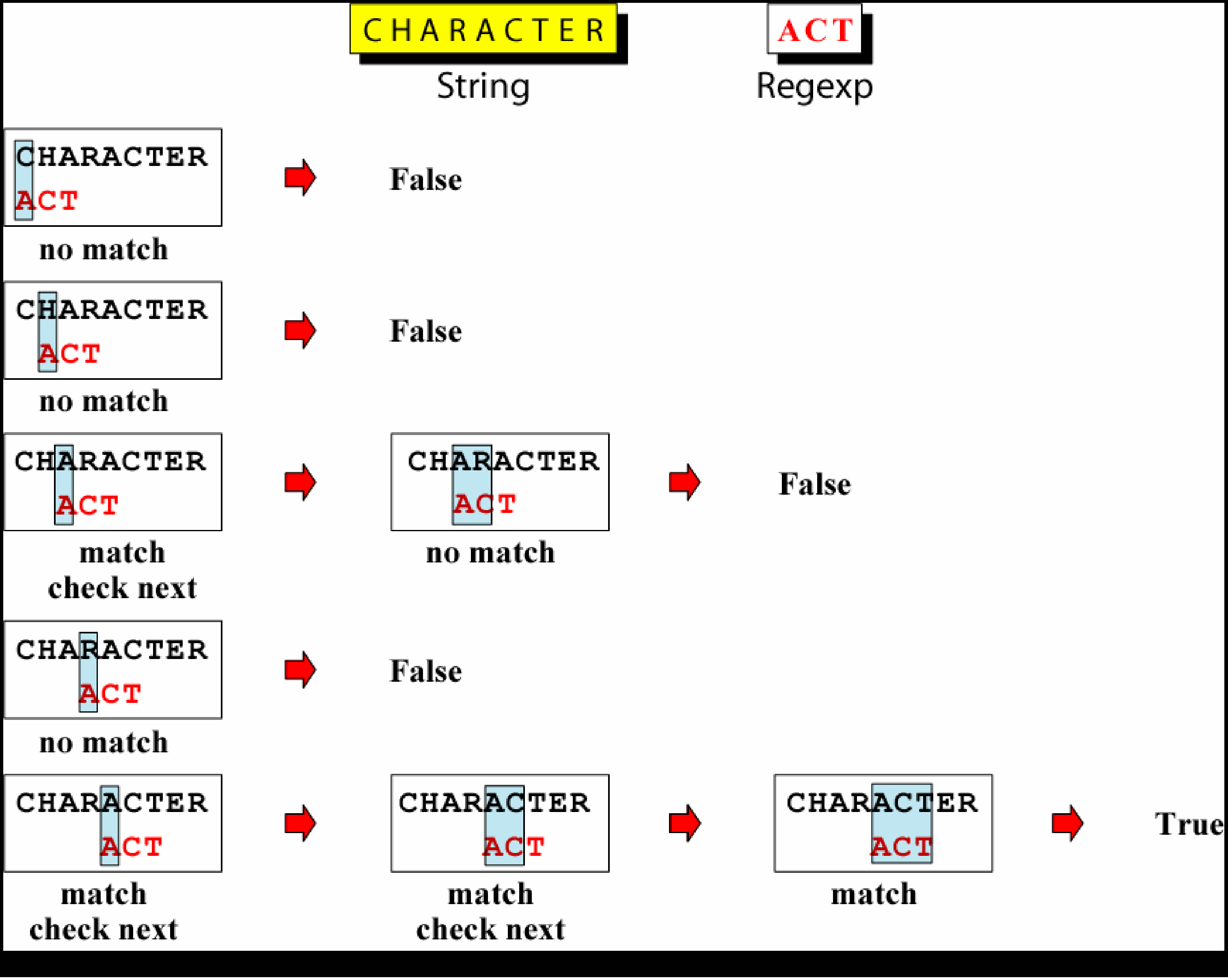
The sequence operator is nothing.

This means that if a series of atoms are shown in a regular expression, it is implied that there is an invisible sequence operator between them.



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

Evaluation of a String Using Sequence Operator



CHARACTER String

ACT Regexp

CHARACTER ACT no match → False

CHARACTER ACT no match → False

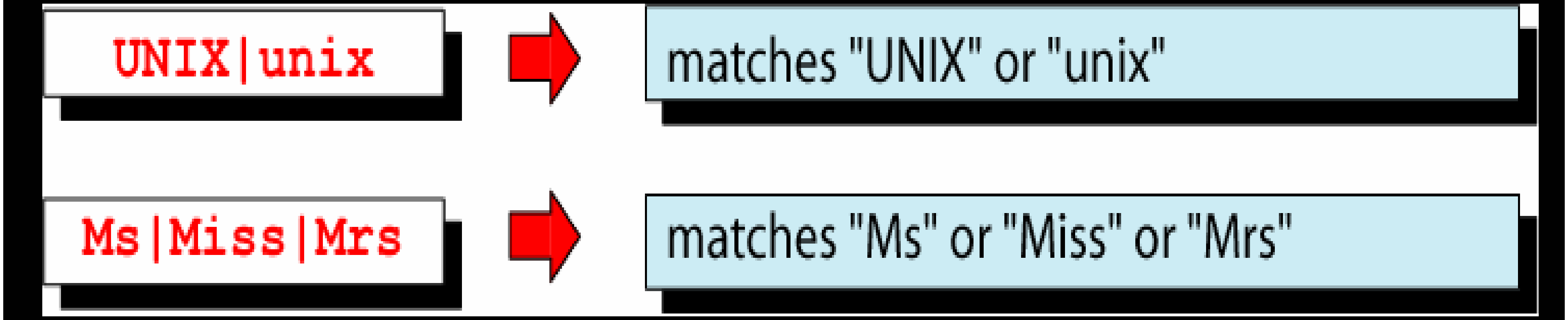
CHARACTER ACT match check next → CHARACTER ACT no match → False

CHARACTER ACT no match → False

CHARACTER ACT match check next → CHARACTER ACT match check next → CHARACTER ACT match → True

Alternation Operator

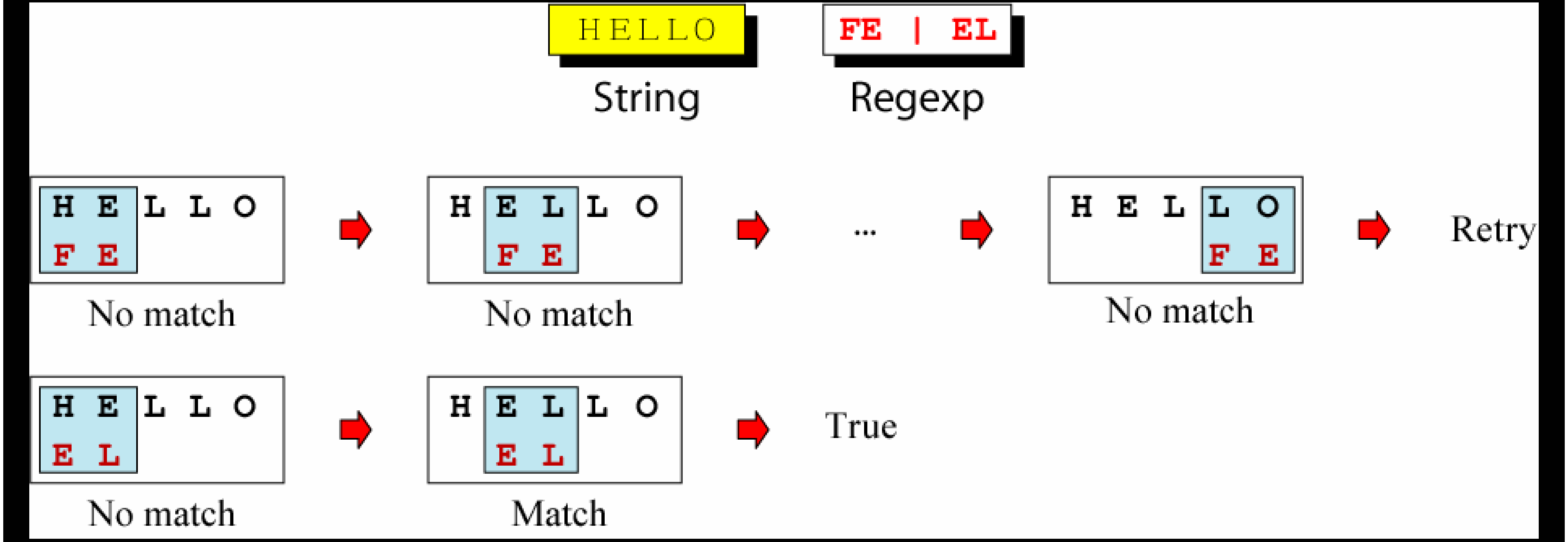
The alternation operator is used to define one or more alternatives.



`UNIX|unix` → matches "UNIX" or "unix"

`Ms|Miss|Mrs` → matches "Ms" or "Miss" or "Mrs"

Matching Alternation Operators

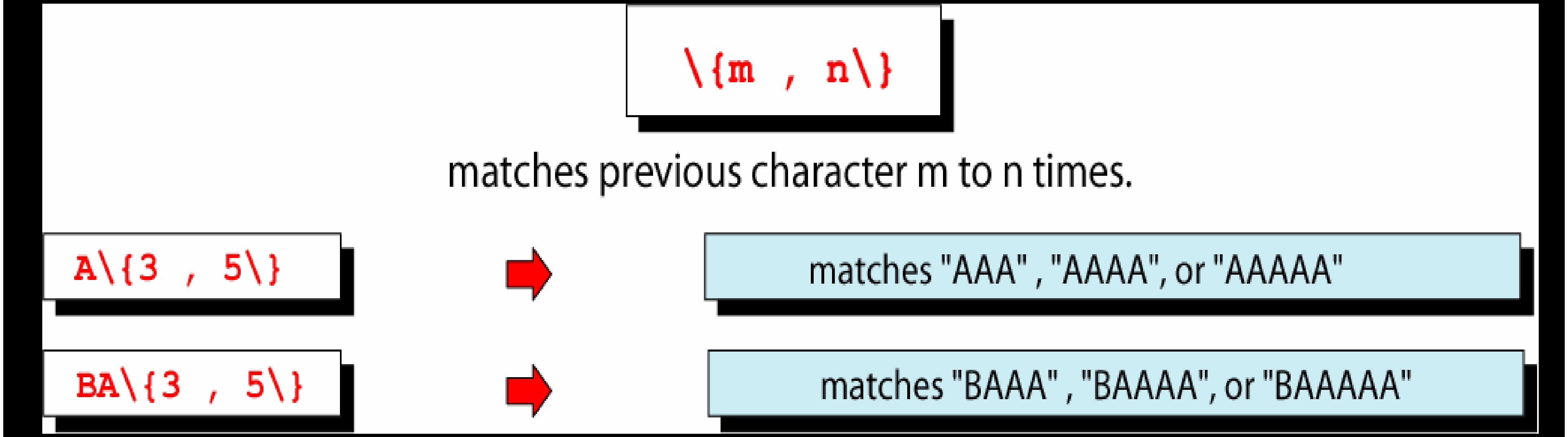


Repetition Operator

The repetition operator specifies that the atom or expression immediately before the repetition may be repeated.

m is a minimum number of repetitions.

n is a maximum number of repetitions.



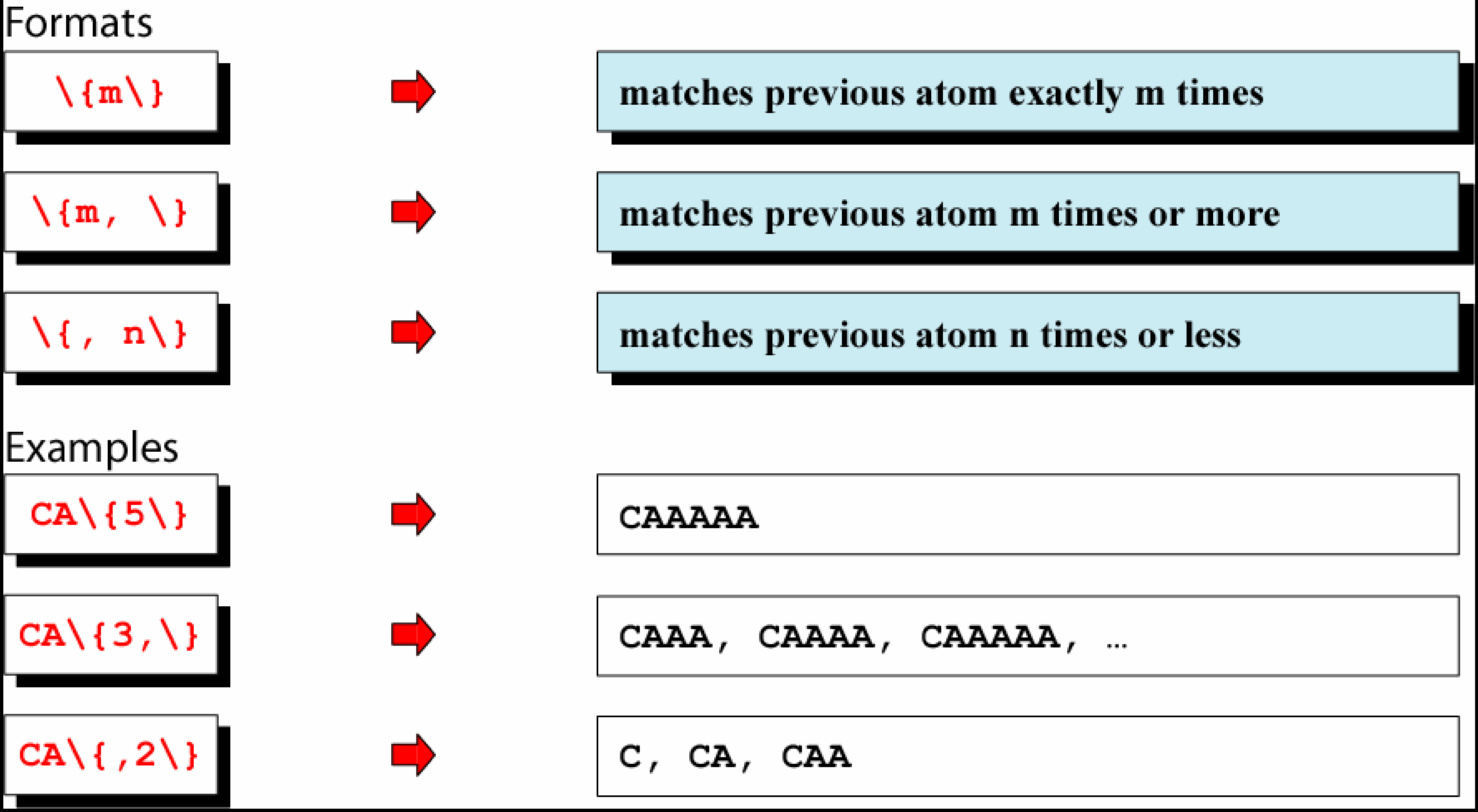
`\{m , n\}`

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



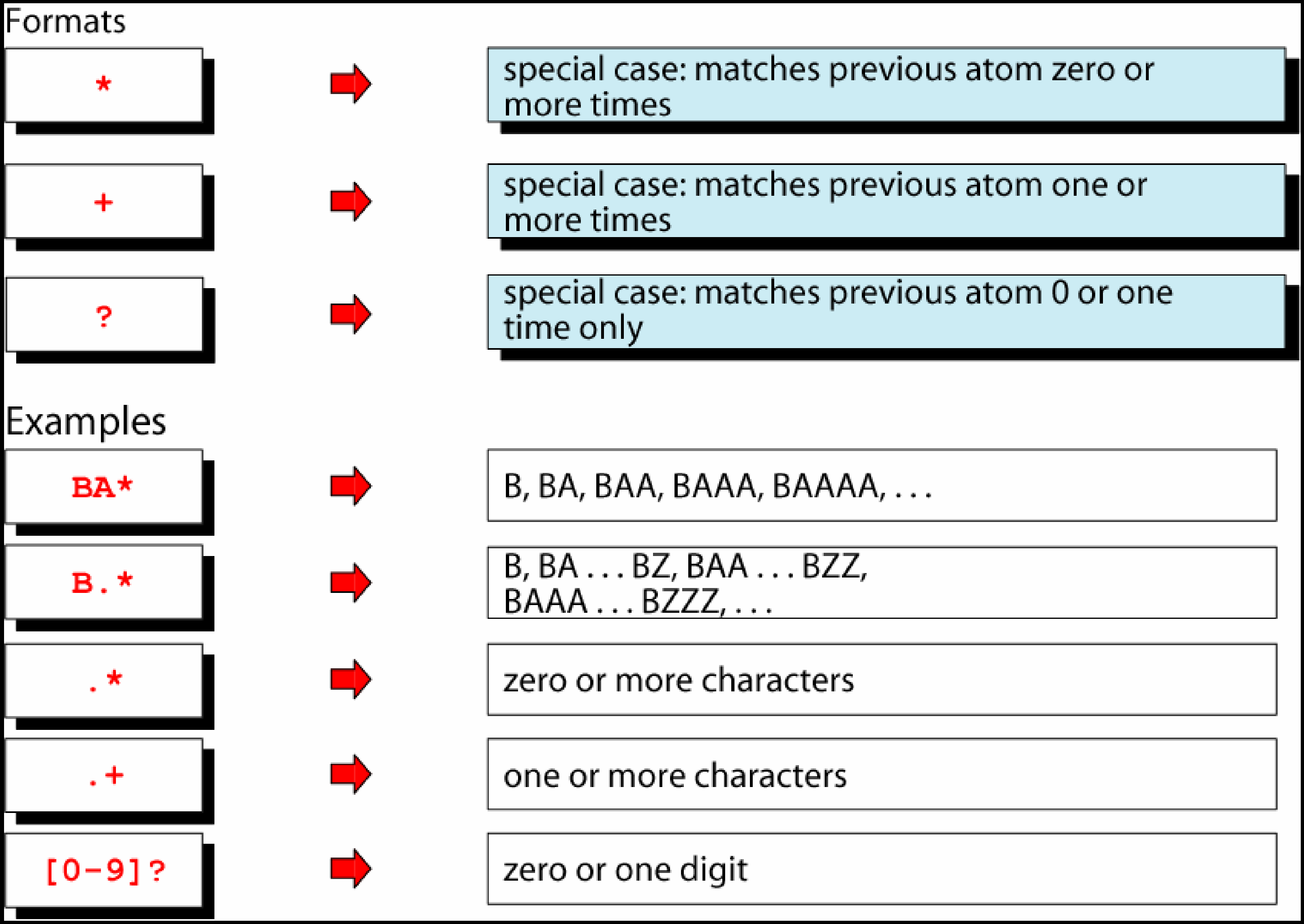
Formats

`\{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

Example of Short Form Repetition Operators



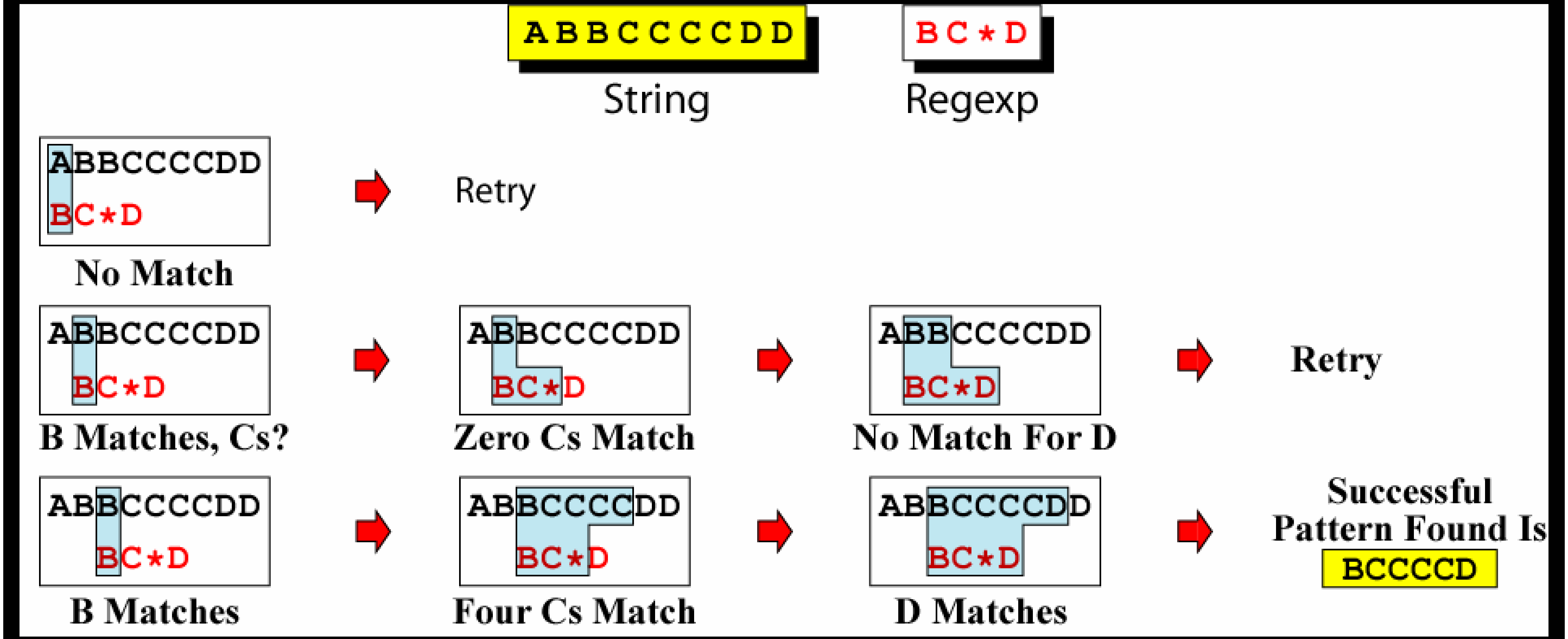
Formats

*	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



ABBCCCCDD

String

BC*D

Regexp

ABBCCCCDD
BC*D
No Match → Retry

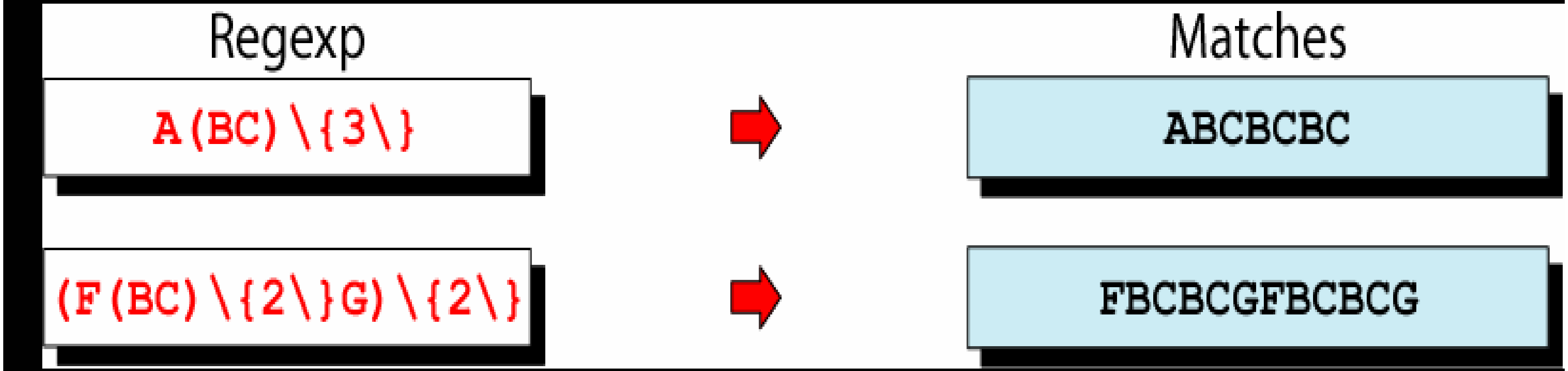
ABBCCCCDD
BC*D
B Matches, Cs? → Zero Cs Match → No Match For D → Retry

ABBCCCCDD
BC*D
B Matches → Four Cs Match → D Matches → Successful Pattern Found Is BCCCCD

Group Operator

The group operator is a pair of opening and closing parentheses.

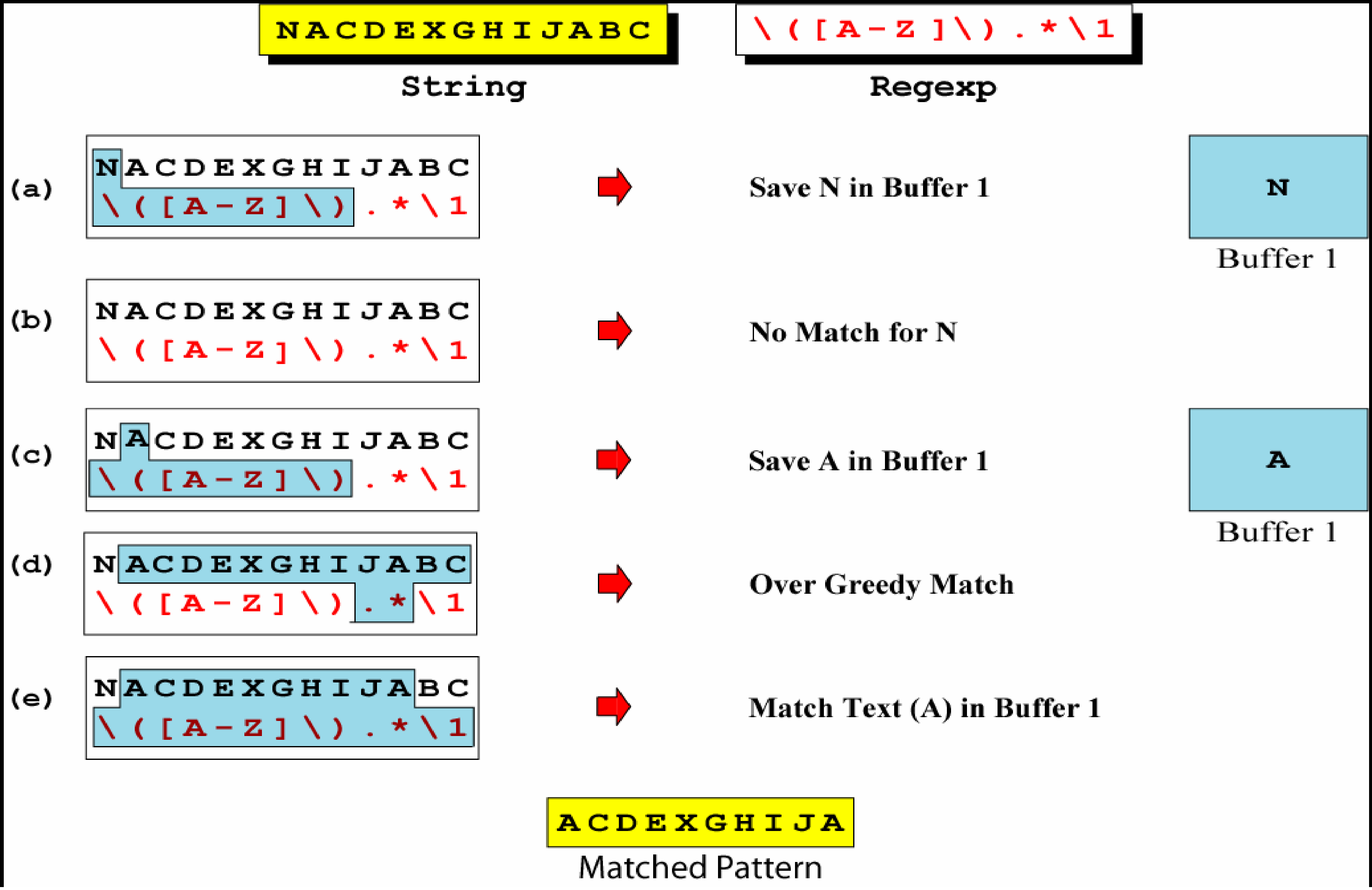
When a group of characters is enclosed in parentheses, the next operator applies to the whole group.



Regexp		Matches
`A(BC)\{3\}`	➡	ABCBCBC
`(F(BC)\{2\}G)\{2\}`	➡	FBCBCGFBCBCG

Saving

The save operator \(   )\ copies a matched text string to one of nine buffers for later reference.



NACDEXGHIJABC

String

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

Regexp

(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