Regular Expressions Made Easy

Operator List

Simple characters match simple characters

/a/ -> a

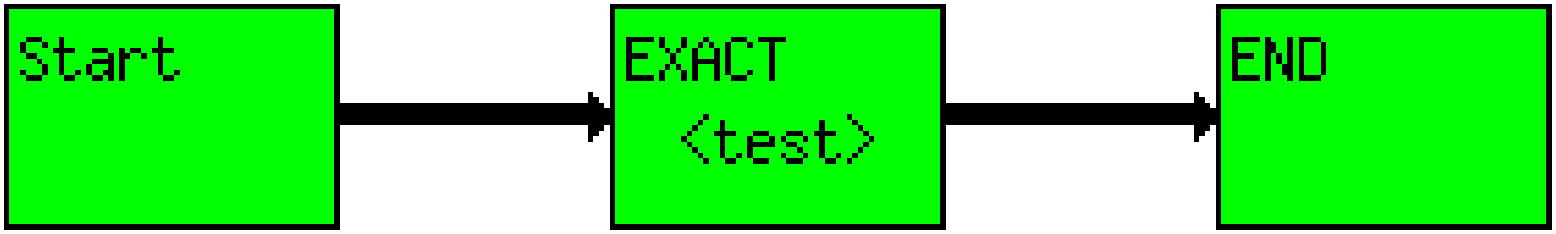
/b/ -> b

/cd/ -> cd

Regular Expression State Machine

1. Match current location in the string against the current node.
2. If it matches, update the string position and move the next node.
3. When coming to a fork, take the top.
4. If the string does not match a node, go back to the previous fork and try the lower branch.
5. If you reach the end the match was successful.

Regular Expression: /test/



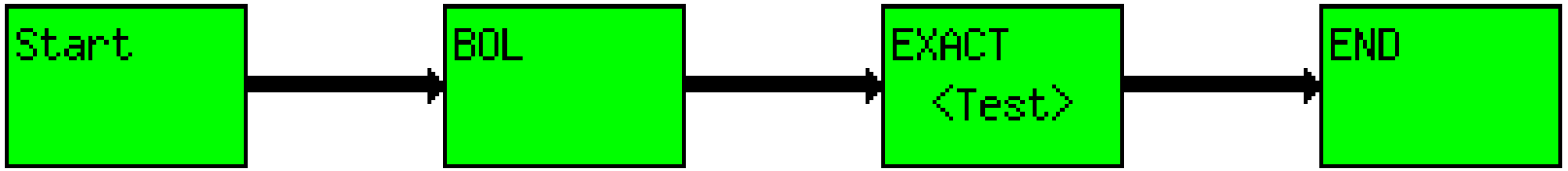
Anchors

^Match the beginning of the line
$ Match the end of the line

a-test	matches	`/test/`
a-test	no match	`/^test/`

Anchor Example

```
Regular Expression: /^Test/
```



Regular Expression Operators

abc… Exact match
^ Line start
$ Line end
* 0 or more time

* -- Match zero or more items

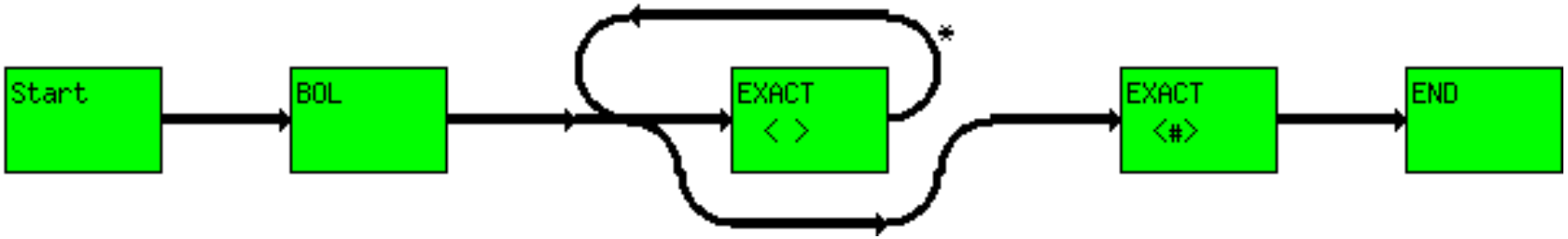
`/a*b/` matches ab, aab, aaab, aaaaab

* is greedy (it matches as much as possible)

Question: Does /a*b/ match “b”?

* Example

Regular Expression: /^ *#/




```
sam
# something
    # something else
```

Greedy Operators (i.e. *)

- * Matches as many characters as possible
- WARNING: Greed can surprise you

What does /a*/ match in the following

```
“test aa of aaa greed “ =~ /a*/
```

1. It matches “aa”, the first match
2. It matches “aaa”, the longer match
3. This is a trick question.

Hint: * = 0 or more

```
“test aa of aaa greed “ =~ /a*/
```

Location of 0 “a”s

Other Common Mistakes

```
$line = “I got spaces  a  lot”;

# Change spaces to underscore
$line =~ s/\s*/_/;
# Wrong

# Split out the words (WRONG)
my @words = split /\s*/, $data;
```

Doing it right

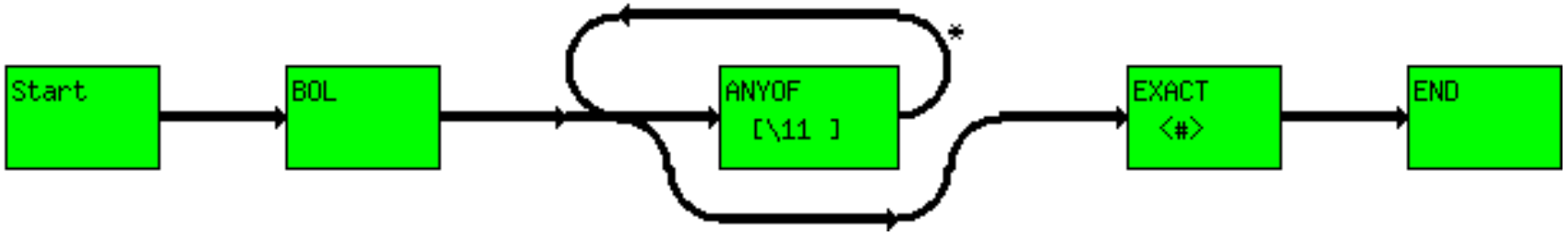
```
$line = “I got spaces  a  lot”;

# Change spaces to underscore
$line =~ s/\s+/_/;
# Right

# Split out the words (Right)
my @words = split /\s+/, $data;
```

Regular Expression Operators

abc...	Exact match
^	Line start
$	Line end
*	0 or more time
[abc]	a or b or c

Example

Regular Expression: /^[\t]*#/



Regular Expression Operators

abc…	Exact match
^	Line start
$	Line end
*	0 or more time
[abc]	a or b or c
[^abc]	not a or b or c
.	Any character
(…)	grouping and \1

Check for repeated characters

```
/(.)\1/
```

Split a line into text comment

Format of the line

text spaces # comment

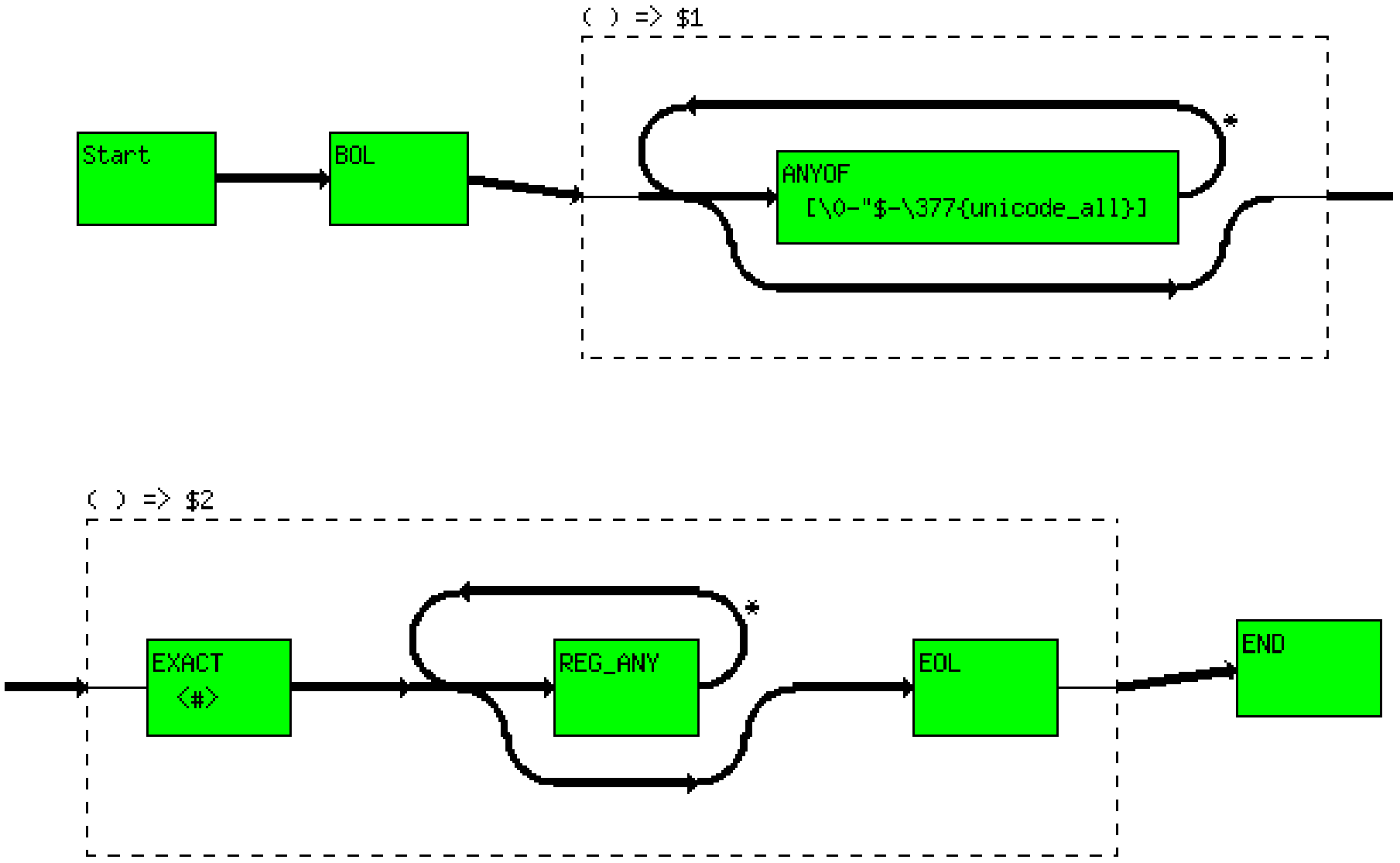
Regular expression

```
/^([^#]*)(#.*$)/
```

Regular Expression Commented

```
 +---------------- Beginning of line
 | ++++---------- Anything except #
 | ||||+--------- 0 or more times
 |+|||||+-------- Put in \1
 |||||||| +------ # (literal)
 |||||||| |++---- Any ch(0 or more times)
 |||||||| |||+--- End of line
 ||||||||+||||+-- Put in \2
/^([^#]*)(#.*$)/
```

Regular Expression: /^([^#]*)(#.*$)/



Regular Expression Operators

abc...	Exact match	a|b	Match a or b
^	Line start		
$	Line end		
*	0 or more time		
[abc]	a or b or c		
[^abc]	not a or b or c		
.	Any character		
(...)	grouping and \1		

The quoted string problem

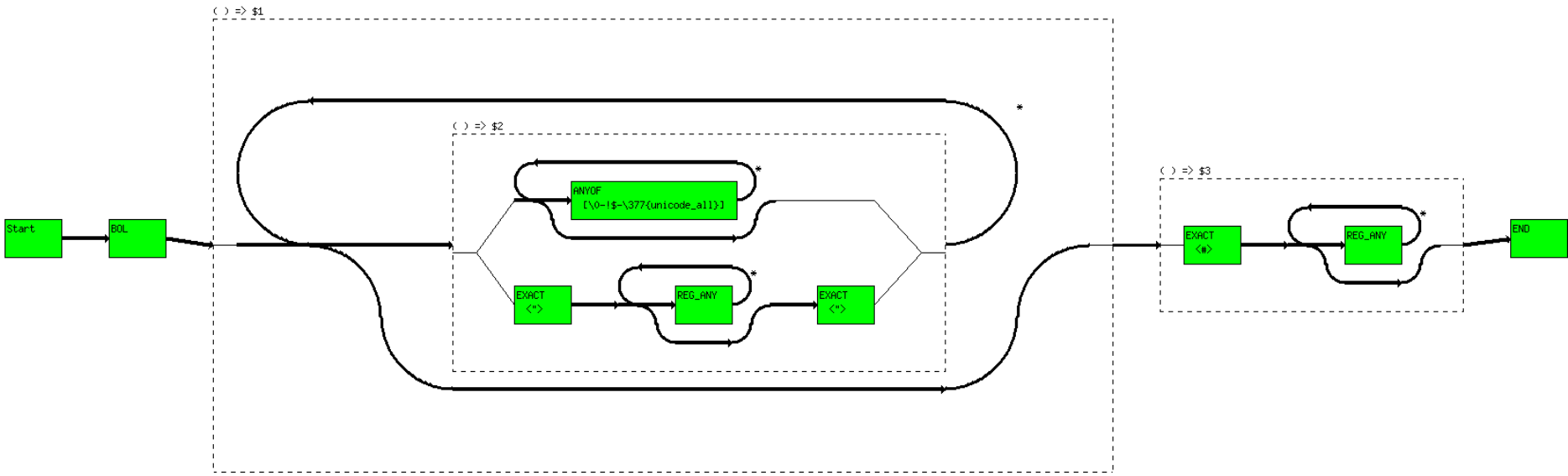
- Match:

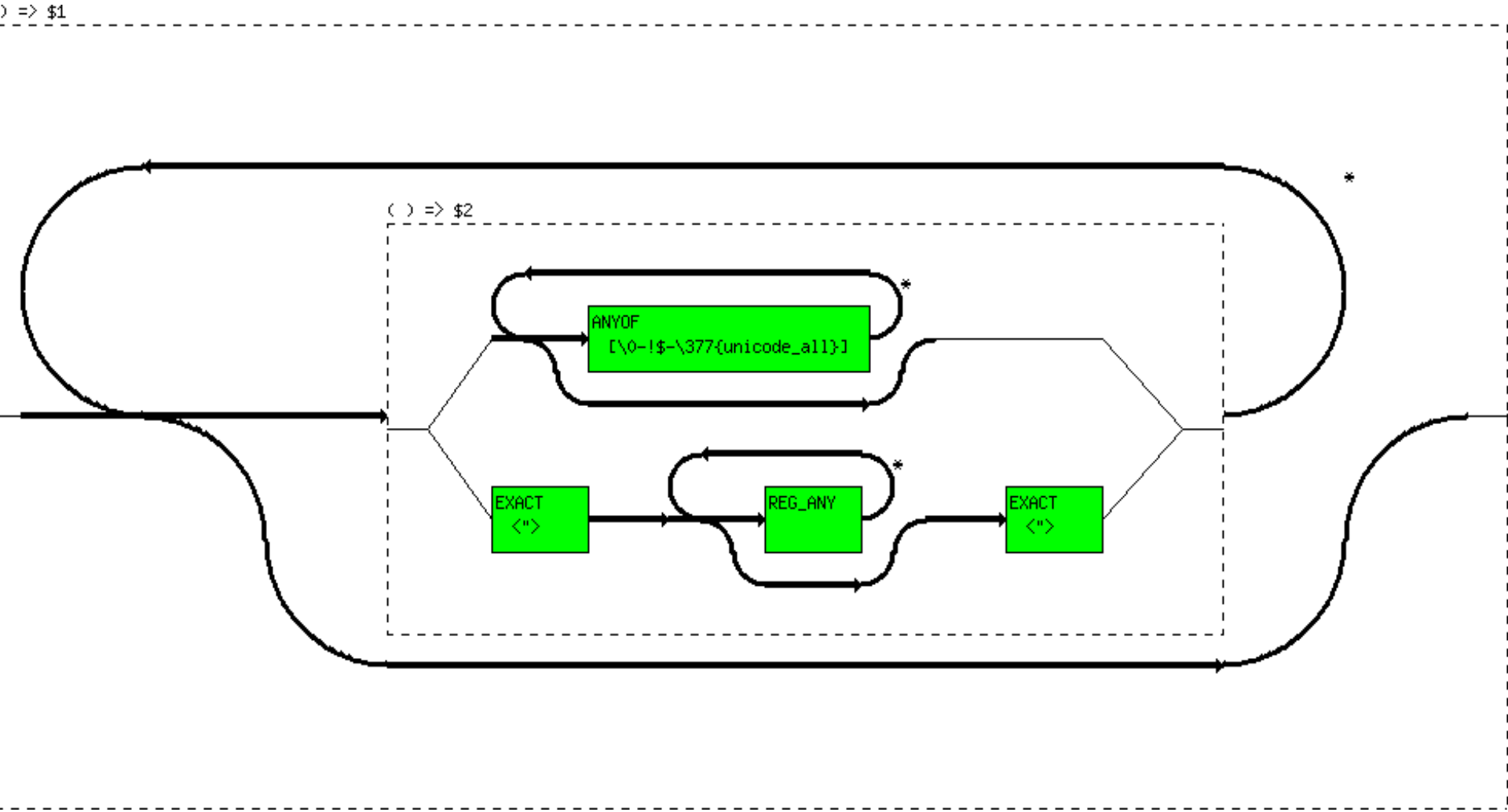
```
test                              # comment
“string # nasty”                  # Comment
```

- New regular expression

```
/^(([^#”]*|”.*”)*)(#.*)/;
```

Regular Expression: /^(([^#"]*|".*")*)(#.*)/



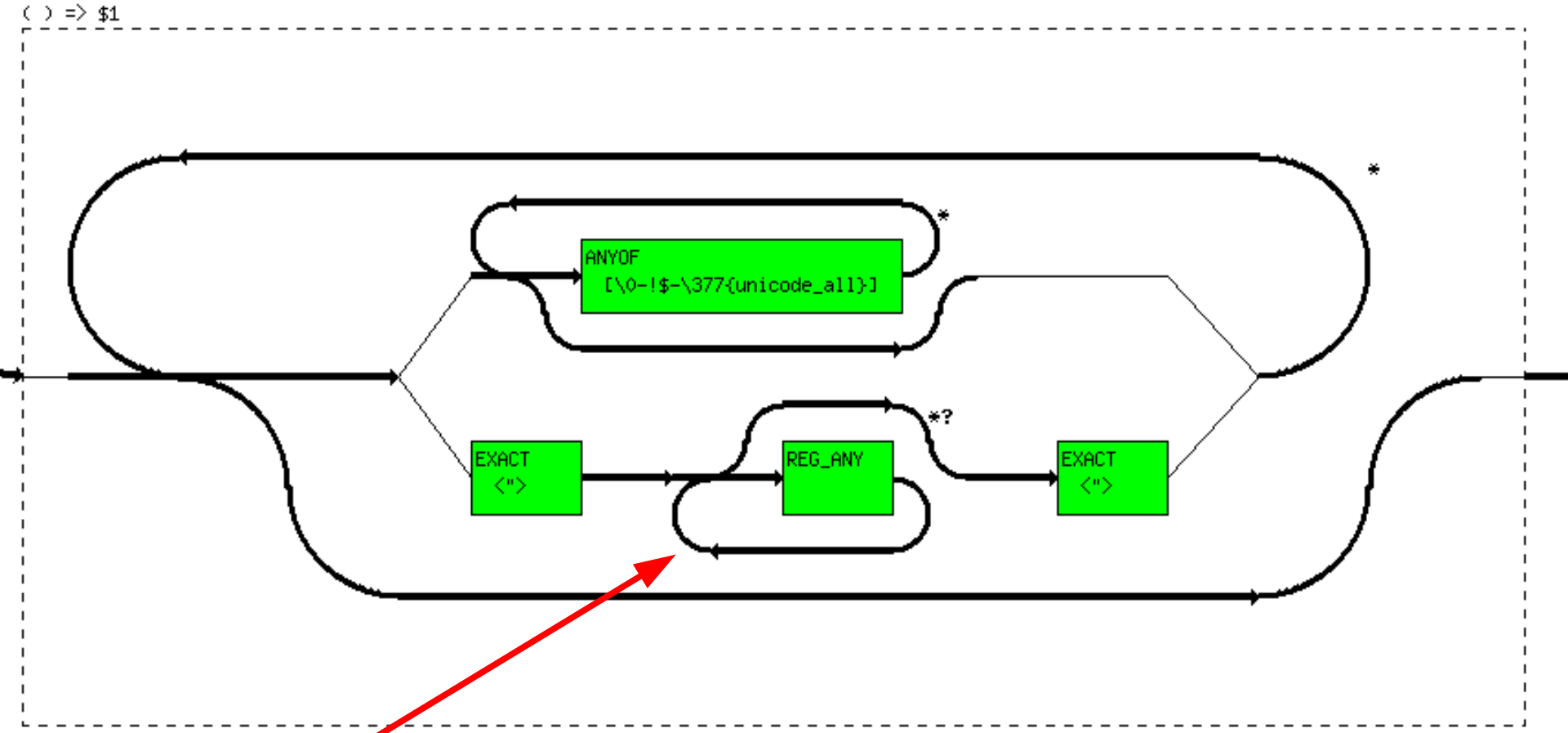

) => $1
( ) => $2
*
*
ANYOF
[\0-!$-\377{unicode_all}]
*
EXACT
<">
REG_ANY
EXACT
<">

Problem

- What about
  `“a string “ # comment has “ inside`
- What's matched
  `“a string “ # comment has “` `inside`

Regular Expression Operators

abc...	Exact match
^	Line start
$	Line end
*	0 or more time
[abc]	a or b or c
[^abc]	not a or b or c
.	Any character
(...)	grouping and \1
a|b	Match a or b
*?	Like * but not greedy



Order has flipped. We try and find a “ first

Problem

- The expression:

```
/^(([^#”]*|”.*”)*)(#.*)/;
```

puts the operator into $1 and the comment into $3 (and junk in $2)

```
/^(([^#”]*|”.*”)*)(#.*)/;
```

Solution

- New operator (?:…). Like () but no \1 or $1.

```
/^((?:[^#”]*|”.*”)*)(#.*)/;
```

Regular Expression Operators

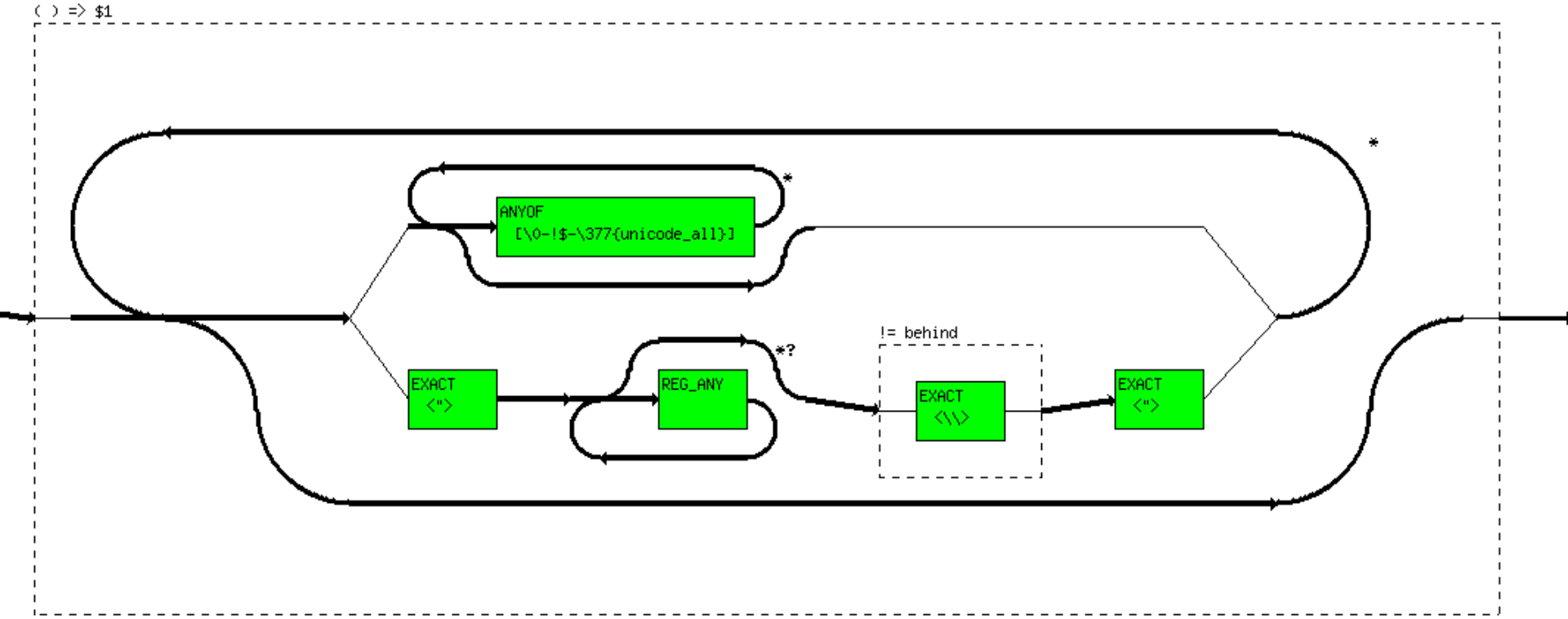
abc…	Exact match
^	Line start
$	Line end
*	0 or more time
[abc]	a or b or c
[^abc]	not a or b or c
.	Any character
(…)	grouping and \1
a|b	Match a or b
*?	Like * but not greedy
(?:…)	Like () but no \1
(?<!x)	Negative look behind. (The previous character can't be an x at this point.)

Final Regular Expression

- Match “xxx” but allow for “xxx\”yyy”

```
/^((?:[^#”]*|”.*(?<!\\)”)*)(#.*)/;
```

Graph



The Challenge

- A regular expression to match mail addresses:

```
^(([^<>()[\]\\.,;:\s@\"]+
(\.[^<>()[\]\\.,;:\s@\"]+)*)|
(\".+\"))@((\[[0-9]{1,3}\.
[0-9]{1,3}\.[0-9]{1,3}\.
[0-9]{1,3}\])|
(([a-zA-Z\-0-9]+\.)+
[a-zA-Z]{2,}))$
```

Graph

Regular Expression: `/^(([^<>()[\]\.,;:\s@"]+(\.[^<>()[\]\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/`