

HW10

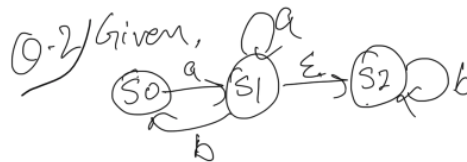
Q1. Give an equivalent Perl regex for the given FSA.

```
my $r = qr/\b(ab)*(aa*b*)\b/;
my $my_str = $ARGV[0];

if ($my_str =~ $r) {
    print("Matched \n");
} else {
    print "Not Matched \n";
}
```

```
HW10 > perl p1.pl a
Matched
HW10 > perl p1.pl b
Not matched
HW10 > perl p1.pl aa
Matched
HW10 > perl p1.pl ab
Matched
HW10 > perl p1.pl abba
Not matched
```

Q2. Convert the NDFSA to a (deterministic) FSA. Draw the machine.



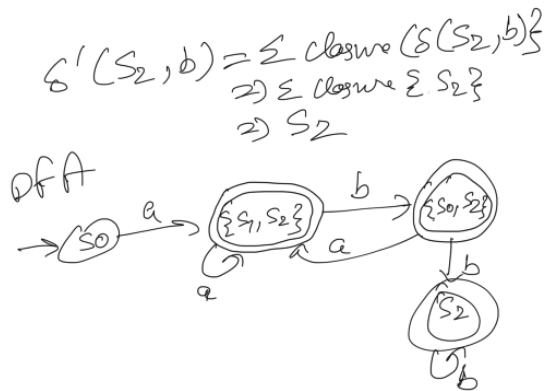
Σ closure of $\{S_0\} = \{S_0\}$
 Σ closure of $\{S_1\} = \{S_1, S_2\}$
 Σ closure of $\{S_2\} = \{S_2\}$

Let Σ closure of S_0 be state S_0
 $S'(S_0, a) = \Sigma \text{ closure } \{S(S_0, a)\}$
 $= \Sigma \text{ closure } \{S_1\}$
 $= \{S_1, S_2\}$
 $= S_B$

$S'(S_0, b) = \Sigma \text{ closure } \{S(S_0, b)\}$
 $= \Sigma \text{ closure } \{\emptyset\}$
 $= \emptyset$

$S'(S_B, a) = \Sigma \text{ closure } \{S(S_1, S_2, a)\}$
 $= \Sigma \text{ closure } \{S(S_1, a), S(S_2, a)\}$

$$\begin{aligned}
&\Rightarrow \Sigma \text{ closure } \{S_1\} \\
&\Rightarrow \{S_1, S_2\} = S_B \\
S'(S_B, b) &= \Sigma \text{ closure } \{\delta(S_1, S_2, b)\} \\
&\Rightarrow \Sigma \text{ closure } \{S_0 \cup S_2\} \\
&\Rightarrow \{S_0, S_2\} \\
&\Rightarrow S_C \\
S'(S_C, a) &= \Sigma \text{ closure } \{\delta(S_0, S_2, a)\} \\
&\Rightarrow \Sigma \text{ closure } \{\delta(S_0, a), \delta(S_2, a)\} \\
&\Rightarrow \Sigma \text{ closure } \{S_1\} \\
&\Rightarrow \{S_1, S_2\} = S_B \\
S'(S_C, b) &= \Sigma \text{ closure } \{\delta(S_0, S_2, b)\} \\
&\Rightarrow \Sigma \text{ closure } \{\delta(S_0, b), \delta(S_2, b)\} \\
&\Rightarrow \Sigma \text{ closure } \{S_2\} \\
&\Rightarrow S_2 \\
S'(S_2, a) &= \Sigma \text{ closure } \{\delta(S_2, a)\} \\
&\Rightarrow \emptyset
\end{aligned}$$



Q3. Give the implementation of the FSA in Perl.

```

%delta = (
    s0 => { a => "sb" },
    sb => { a => "sb", b => "sc"},
    sc => { a => "sb", b => "s2"},
    s2 => { b => "s2"}
);
$state = "s0";
foreach $c (split (//, @ARGV[0])) {
    $state = $delta{$state}{$c};
}

```

```

}
print(($state eq "sb" or $state eq "sc" or $state eq "s2" ) ? "Accept\n" : "Reject\n")

```

Output:

```

● HW10 > perl p3.pl a
Accept
● HW10 > perl p3.pl b
Reject
● HW10 > perl p3.pl aa
Accept
● HW10 > perl p3.pl ab
Accept
● HW10 > perl p3.pl abba
Reject

```

Q4. Run your two Perl programs and give examples:

- a, *b, aa, ab, *ba, aaab, abaabb, *abba, *abaabbbaaabb

```

● HW10 > perl p1.pl a
Matched
● HW10 > perl p1.pl b
Not Matched
● HW10 > perl p1.pl aa
Matched
● HW10 > perl p1.pl ab
Matched
● HW10 > perl p1.pl ba
Not Matched
● HW10 > perl p1.pl aaab
Matched
● HW10 > perl p1.pl abaabb
Matched
● HW10 > perl p1.pl abba
Not Matched
● HW10 > perl p1.pl abaabbbaaabb
Not Matched

```

```

● HW10 > perl p3.pl a
Accept
● HW10 > perl p3.pl b
Reject
● HW10 > perl p3.pl aa
Accept
● HW10 > perl p3.pl ab
Accept
● HW10 > perl p3.pl ba
Reject
● HW10 > perl p3.pl aaab
Accept
● HW10 > perl p3.pl abaabb
Accept
● HW10 > perl p3.pl abba
Reject
● HW10 > perl p3.pl abaabbbaaabb
Reject

```