

# From Regular Expressions To Deterministic Automata

**Paper**

Gerard Berry Ravi Sethi

**Implementation & Presentation**

Corey Lear Ken Gorab Tom Hrabchak

# Paper Overview

- A new algorithm to go from regular expressions to DFAs
- Refines Brzozowski's elegant algorithm into McNaughton and Yamada's faster algorithm

# Paper Overview

## Brozowski's construction of a DFA

Input: Regular Expression  $E$

Output: DFA  $D$

1. The states of  $D$  are the distinct derivatives  $w^{-1}E$ , for all strings  $w$
2. Construct a transition under symbol  $a$  from state  $p$  to state  $q$  if and only if  $p$  is for derivative  $w^{-1}E$ , for some  $w$ , and  $q$  is for  $[(wa)^{-1}]E$
3. The state for  $E$  is the start state.
4. A state is an accepting state if and only if it is for a derivative  $w^{-1}E$ , for some  $w$ , and  $\delta([w^{-1}]E) = 1$ ; that is the empty string is in  $[L(wa)^{-1}]E$ .

## McNaughton and Yamada's construction of a DFA

Input: Marked Regular Expression  $E'$

Output: DFA  $M'$

1.  $M'$  has a state for the continuation of each marked symbol in  $E'$
2. Construct a transition under symbol  $a$  from state  $p$  to state  $q$  for the continuation of  $a$  if and only if  $p$  is for some continuation  $C$  and  $C$  can generate a string with a leading  $a$
3. The state for the entire expression  $E'$  is the start state.
4. A state is an accepting state if and only if it is for a continuation  $C$  and  $\delta(C)=1$

# Paper Overview

## Brozowski's construction of a DFA

Input: Regular Expression  $E$

Output: DFA  $D$

- Introduces the derivative and delta of a regular expression

## McNaughton and Yamada's construction of a DFA

Input: Marked Regular Expression  $E'$

Output: DFA  $M'$

- Introduces marked regular expressions

## New Algorithm

Input: Marked Regular Expression  $E'$

Output: DFA  $M'$

- Constructs transitions using  $first(E')$  and  $follow(b)$ , where  $b$  is a state
- Accepting states are found by testing if a special end character is in  $follow(b)$

# Take away from Brzozowski

## Delta Notation

- $\delta(E)$  stands for 1 if  $L(E)$  contains the empty string
- $\delta(E)$  is computed recursively on  $E$ 
  - $\delta(0) = 0$
  - $\delta(1) = 1$  (where 1 is taken to mean  $\varepsilon$ )
  - $\delta(E_1 \cup E_2) = \delta(E_1) + \delta(E_2)$
  - $\delta(E_1 E_2) = \delta(E_1) \cdot \delta(E_2)$
  - $\delta(E^*) = 1$
- It is clear then that  $\delta(E)$  represents the number of possibilities the expression has of containing the empty string  $\varepsilon$ .

# Take aways from McNaughton and Yamada

## Marked Expressions

- Mark all input symbols in a regular expression
  - $(ab \cup b)^*ba$  becomes  $(a_1b_2 \cup b_3)^*b_4a_5$
- These subscripted symbols are distinct and will become the states of an automaton

## First

- $first(E) = \{a : av \in L(E)\}$
- The set of possible first symbols in the language of a regular expression  $E$

## Follow

- $follow_E(a) = \{b : uabv \in L(E)\}$
- The set of possible symbols that come immediately after the symbol  $a$  in the language of  $E$
- For a marked symbol  $a_1$ ,  $follow(a_1)$  will contain the set of marked symbols corresponding to the next possible states in the automaton

# New Algorithm

Input: Marked Regular Expression  $E'$

Output: DFA  $M'$

1.  $M'$  has a start state plus a state for each marked symbol  $a$  in  $E'$
2. Construct a transition from the start state to the state for  $a$  if and only if  $a$  is in  $first(E')$
3. Construct a transition from the state  $b$  to the state  $a$  if and only if  $a$  is in  $follows(b)$
4. The start state is an accepting state if and only if  $delta(E')=1$
5. The state for  $a$  is an accepting state if and only if  $!$  is in  $follows(a)$

# General Idea of Using New Algorithm

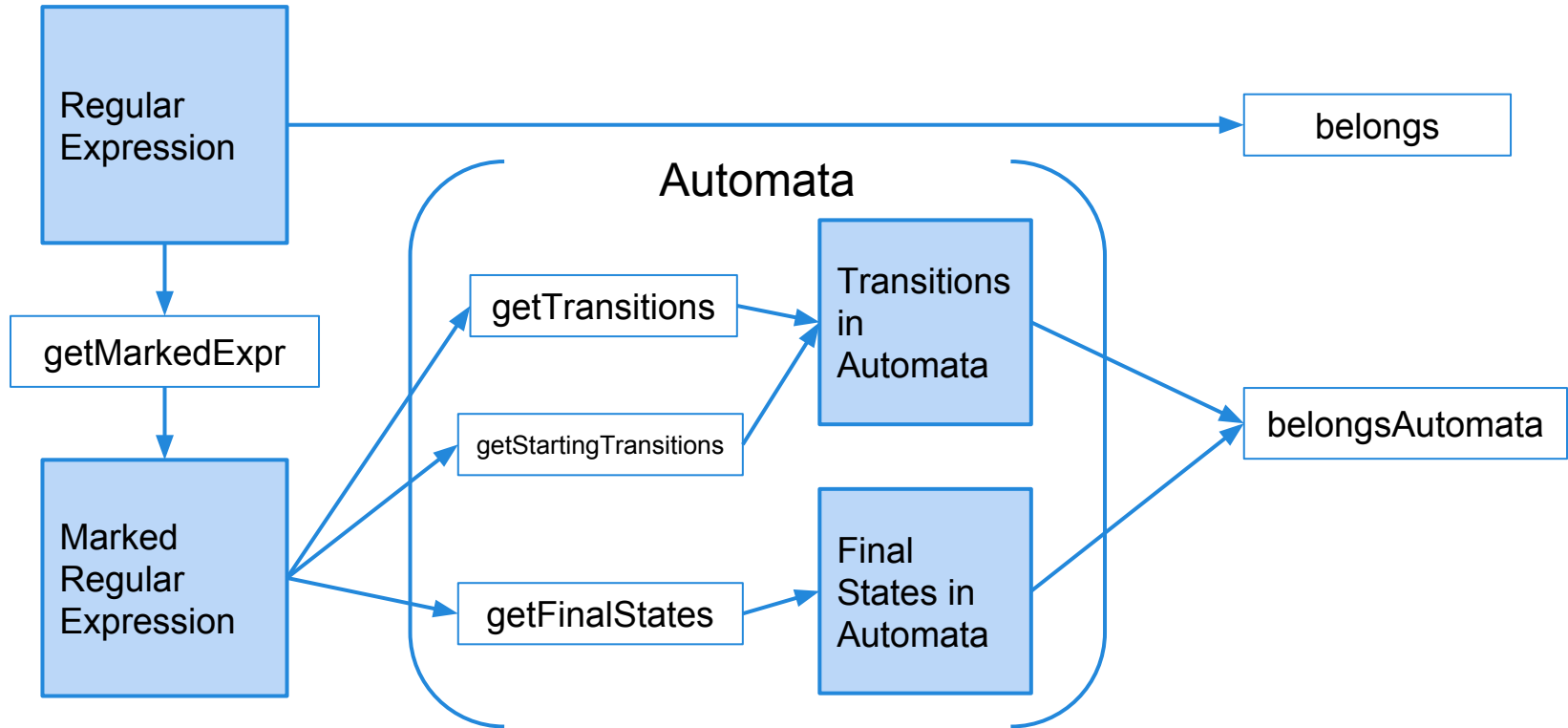
1. Construct an automata from a marked expression
2. The marks on the transitions are then erased, resulting in an NFA for the original unmarked expression
  - Note: This approach works for the usual operations of union, concatenation, and iteration, but does not work with intersection and complement. This is because marking and unmarking do not preserve the language generated by regular expressions with these operations



# Project Implementation Overview

- We originally intended to implement all 3 algorithms, however we only implemented the new algorithm

# What we did implement



# Implementation

- We used OCaml's algebraic types extensively.
  - Regex, consisting of a char, a Union, a Concatenation, a Star, or Epsilon.
  - These are all custom datatypes that use Cartesian product with regex.

```
type regex =  
  | Eps  
  | Lit of char  
  | Union of regex * regex  
  | Concat of regex * regex  
  | Star of regex;;
```

```
type markedRegex =  
  | MEps  
  | MLit of char * int  
  | MUnion of markedRegex * markedRegex  
  | MConcat of markedRegex * markedRegex  
  | MStar of markedRegex;;
```

# Implementation

## Regular Expression Functions

(marked & unmarked)

- belongs
  - In: string and expr
  - Out: is the string accepted by expr
- delta
  - In: expr
  - Out: *delta(expr)* --> the potential for a string in the expr to include epsilon
- follow
  - In: expr and symbol
  - Out: The set of possible symbols that come immediately after the symbol *a* in the language of *E*
- first
  - In: expr
  - Out: a list of starting symbols in the expression
- last
  - In: expr
  - Out: a list of ending symbols in the expression

# Implementation

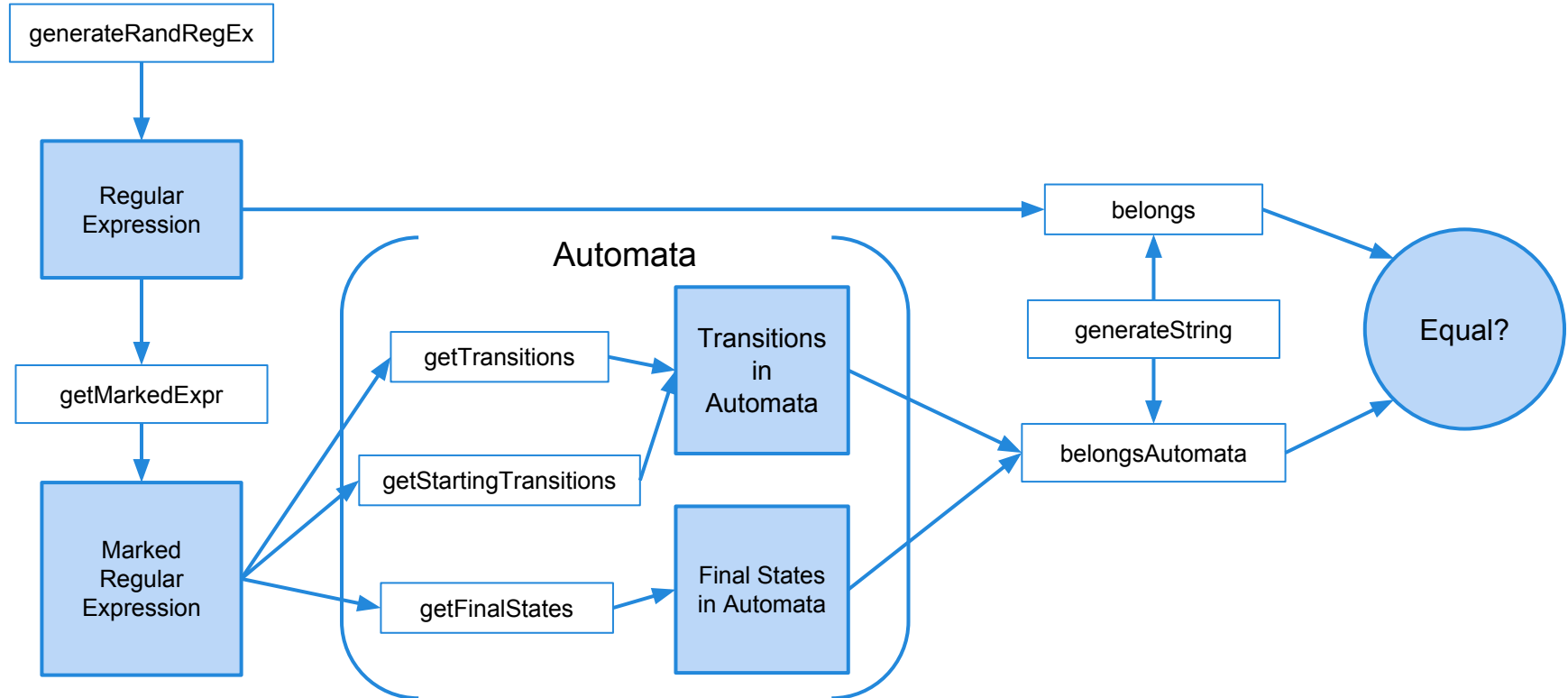
## Automata

- belongs:
  - In: string, transitionlist, finalstates and currentstate
  - Out: is string accepted by automata
- getStates:
  - In: expr
  - Out: states
- getTransitions:
  - In: expr
  - calls two helper functions  
getTransitionsFromStates expr  
getStates
  - Out: prepares a list of transitions
- getfinalsates:
  - In: expr and states
  - Out: accepting states
- getNextStates:
  - In: transitionLists and currentstate
  - Out: possible next states
- getStartingTransitions:
  - In: expr and startingStates
  - Out: transitions from first to next state

# Testing the Implementation

- Created a function test
  - make an alphabet
  - randomly generates
    - string
    - regex
  - makes automata from regex
  - calls `belong` on regex and automata
  - compares the two

# Testing the Implementation



# Script

- Script runs endlessly concatenating output to a file in which we grep errors.

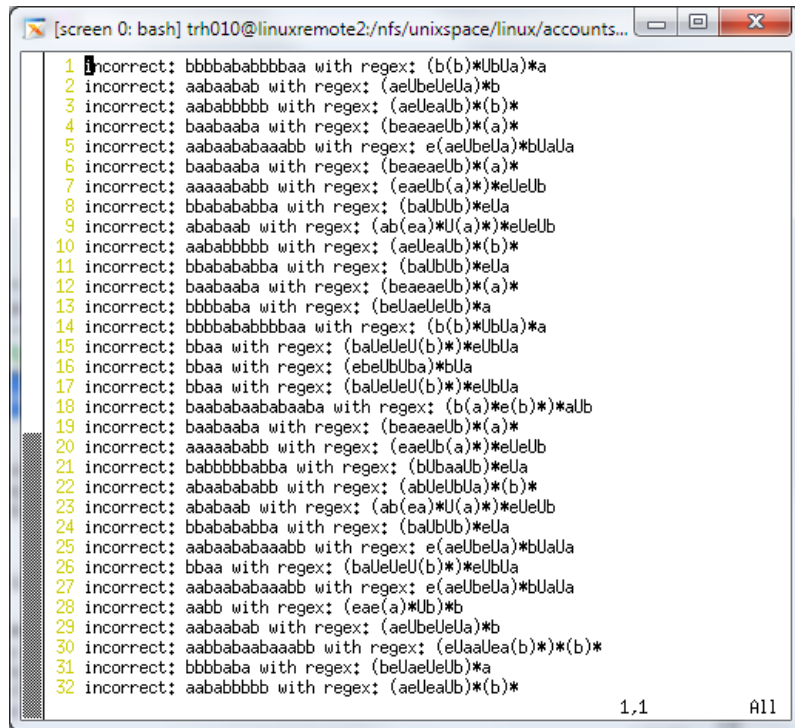
```
#!/bin/bash
echo running script
while true
do
    ./proj_phase3.mlo >> output.txt 2>> errors.txt
done
exit
```

```
#!/bin/bash
cat output.txt | grep "incorrect"
exit
```



# Results

- Test cases: 25,107,109  
(actually still running...)
  - generated regular expressions are not minimized
- Incorrect cases: 32
- Accuracy: 99.999872546%
- So we made a small mistake somewhere...



```
[screen 0: bash] trh010@linuxremote2:/nfs/unixspace/linux/accounts...  
1 incorrect: bbbbababbbbaa with regex: (b(b)*UbUa)*a  
2 incorrect: aabaabab with regex: (aeUbeUeUa)*b  
3 incorrect: aababbbbbb with regex: (aeUeaUb)*(b)*  
4 incorrect: baabaaba with regex: (beaeaeUb)*(a)*  
5 incorrect: aabaababaaabb with regex: e(aeUbeUa)*bUaUa  
6 incorrect: baabaaba with regex: (beaeaeUb)*(a)*  
7 incorrect: aaaaababb with regex: (eaeUb(a)*)*eUeUb  
8 incorrect: bbabababba with regex: (baUUb)*eUa  
9 incorrect: ababaab with regex: (ab(ea)*U(a)*)*eUeUb  
10 incorrect: aababbbbbb with regex: (aeUeaUb)*(b)*  
11 incorrect: bbabababba with regex: (baUUb)*eUa  
12 incorrect: baabaaba with regex: (beaeaeUb)*(a)*  
13 incorrect: bbbababa with regex: (beUaeUeUb)*a  
14 incorrect: bbbbababbbbaa with regex: (b(b)*UbUa)*a  
15 incorrect: bbaa with regex: (baUeUeU(b)*)*eUbUa  
16 incorrect: bbaa with regex: (eUeUbUa)*bUa  
17 incorrect: bbaa with regex: (baUeUeU(b)*)*eUbUa  
18 incorrect: baabaabaabaaba with regex: (b(a)*e(b)*)*aUb  
19 incorrect: baabaaba with regex: (beaeaeUb)*(a)*  
20 incorrect: aaaaababb with regex: (eaeUb(a)*)*eUeUb  
21 incorrect: babbababba with regex: (bUbaaUb)*eUa  
22 incorrect: abaabababb with regex: (aUeUeUa)*(b)*  
23 incorrect: ababaab with regex: (ab(ea)*U(a)*)*eUeUb  
24 incorrect: bbabababba with regex: (baUUb)*eUa  
25 incorrect: aabaababaaabb with regex: e(aeUbeUa)*bUaUa  
26 incorrect: bbaa with regex: (baUeUeU(b)*)*eUbUa  
27 incorrect: aabaababaaabb with regex: e(aeUbeUa)*bUaUa  
28 incorrect: aabb with regex: (eae(a)*Ub)*b  
29 incorrect: aabaabab with regex: (aeUbeUeUa)*b  
30 incorrect: aabbababaaabb with regex: (eUaUea(b)*)*(b)*  
31 incorrect: bbbababa with regex: (beUaeUeUb)*a  
32 incorrect: aababbbbbb with regex: (aeUeaUb)*(b)*
```

# We have a lot correct though...

```
[screen 0: bash] trh010@linuxremote2:/nfs/unixspace/linux/accounts...
1234698 correct: bbba with regex: (b)*
1234699 correct: aaabb with regex: (b)*
1234700 correct: abbabbababbaba with regex: (b)*
1234701 correct: b with regex: (b)*
1234702 correct: bbaababbbaa with regex: (b)*
1234703 correct: baabbaaba with regex: (b)*
1234704 correct: aaabbbbab with regex: (b)*
1234705 correct: bbbabbbbaab with regex: (b)*
1234706 correct: bbabbba with regex: (b)*
1234707 correct: baaabbbbaa with regex: (b)*
1234708 correct: aababbbbabbba with regex: (b)*
1234709 correct: aaab with regex: (b)*
1234710 correct: abbbaab with regex: (b)*
1234711 correct: abbabaaba with regex: (b)*
1234712 correct: babbb with regex: (b)*
1234713 correct: bbab with regex: (bUbUbbUeb)*
1234714 correct: bababaaababba with regex: (bUbUbbUeb)*
1234715 correct: abaaaaabaab with regex: (bUbUbbUeb)*
1234716 correct: babbbab with regex: (bUbUbbUeb)*
1234717 correct: baaabb with regex: aUbUeUaUe(eea)*
1234718 correct: bbaabbaabababb with regex: aUbUeUaUe(eea)*
1234719 correct: bb with regex: aUbUeUaUe(eea)*
1234720 correct: bbaababbbba with regex: aUbUeUaUe(eea)*
1234721 correct: abaaaa with regex: aUbUeUaUe(eea)*
1234722 correct: a with regex: aUbUeUaUe(eea)*
1234723 correct: abbbaa with regex: aUbUeUaUe(eea)*
1234724 correct: a with regex: aUbUeUaUe(eea)*
1234725 correct: bbaabbaabbaaab with regex: aUbUeUaUe(eea)*
1234726 correct: bbbaaaaaababa with regex: aUbUeUaUe(eea)*
1234727 correct: bbbbbbba with regex: aUbUeUaUe(eea)*
1234728 correct: baaab with regex: aUbUeUaUe(eea)*
1234729 correct: abaab with regex: aUbUeUaUe(eea)*
```

1234729,1 4%

```
[screen 0: bash] trh010@linuxremote2:/nfs/unixspace/linux/accounts...
9999947 correct: aabbbbbaabaaa with regex: a
9999948 correct: bbaaabbbbabbbab with regex: a
9999949 correct: aabbbbbaabaa with regex: a
9999950 correct: abbbaabababa with regex: a
9999951 correct: babbb with regex: a
9999952 correct: aa with regex: a
9999953 correct: baabbbbaaaabba with regex: a
9999954 correct: ababbbbab with regex: a
9999955 correct: bbaaaababbbbaab with regex: a
9999956 correct: bbaabbbb with regex: a
9999957 correct: bab with regex: a
9999958 correct: aa with regex: a
9999959 correct: baaabbaaaabab with regex: a
9999960 correct: aabab with regex: (a)*b
9999961 correct: bab with regex: (a)*b
9999962 correct: babababab with regex: (a)*b
9999963 correct: baaab with regex: (a)*b
9999964 correct: babaaaaa with regex: (a)*b
9999965 correct: abaaaaaaa with regex: eUb
9999966 correct: bbbb with regex: eUb
9999967 correct: aaababbaababba with regex: eUb
9999968 correct: bbaaababbaa with regex: eUb
9999969 correct: bbaaaabaaabbbba with regex: eUb
9999970 correct: babbbbaa with regex: eUb
9999971 correct: abb with regex: eUb
9999972 correct: abb with regex: eUb
9999973 correct: a with regex: eUb
9999974 correct: aaaabaab with regex: eUb
9999975 correct: baaabababbaabb with regex: eUb
9999976 correct: bbbbbbbaaababb with regex: eUb
9999977 correct: aabbbba with regex: eUb
9999978 correct: aab with regex: eUb
```

9999947,1 38%

# and more...

```
[screen 0: bash] trh010@linuxremote2:/nfs/unixspace/linux/accounts...
4999974 correct: baaab with regex: (b)*UeUbeUbeUa
4999975 correct: bbabbbba with regex: (b)*UeUbeUbeUa
4999976 correct: abababbaab with regex: (b)*UeUbeUbeUa
4999977 correct: aaab with regex: (b)*UeUbeUbeUa
4999978 correct: abaaaa with regex: (b)*UeUbeUbeUa
4999979 correct: ababbbabbb with regex: (b)*UeUbeUbeUa
4999980 correct: aabbb with regex: (b)*UeUbeUbeUa
4999981 correct: ab with regex: (b)*UeUbeUbeUa
4999982 correct: aabaaabba with regex: (b)*UeUbeUbeUa
4999983 correct: bbabaa with regex: (b)*UeUbeUbeUa
4999984 correct: a with regex: (b)*UeUbeUbeUa
4999985 correct: aaabaaabaaaaa with regex: (b)*UeUbeUbeUa
4999986 correct: abbaab with regex: (b)*UeUbeUbeUa
4999987 correct: abaababbbab with regex: (b)*UeUbeUbeUa
4999988 correct: aabbbabbbbbb with regex: (b)*UeUbeUbeUa
4999989 correct: bbaabaaababa with regex: (b)*UeUbeUbeUa
4999990 correct: baabaaa with regex: (b)*UeUbeUbeUa
4999991 correct: baaababaa with regex: eUeUaUeUa
4999992 correct: baaababbaaa with regex: eUeUaUeUa
4999993 correct: bbabab with regex: eUeUaUeUa
4999994 correct: babbbbbb with regex: eUeUaUeUa
4999995 correct: abaaaaabbaaaa with regex: eUeUaUeUa
4999996 correct: bbbbaaabbaaaaa with regex: eUeUaUeUa
4999997 correct: baaababaaa with regex: eUeUaUeUa
4999998 correct: aaabbaabbbaba with regex: eUeUaUeUa
4999999 correct: babaa with regex: eUeUaUeUa
5000000 correct: bbabb with regex: eUeUaUeUa
5000001 correct: ab with regex: eUeUaUeUa
5000002 correct: abba with regex: eUeUaUeUa
5000003 correct: bbaaaaaaab with regex: eUeUaUeUa
5000004 correct: bbaabbaababbbb with regex: eUeUaUeUa
5000005 correct: a with regex: eUeUaUeUa
4999974,1 19%
```

```
[screen 0: bash] trh010@linuxremote2:/nfs/unixspace/linux/accounts...
14999985 correct: aabbbbbbbabb with regex: bUaUa
14999986 correct: baaaabbbb with regex: bUaUa
14999987 correct: bbbabbbbbbb with regex: bUaUa
14999988 correct: bbbbaabba with regex: bUaUa
14999989 correct: baaabab with regex: bUaUa
14999990 correct: ab with regex: bUaUa
14999991 correct: bbbababbb with regex: bUaUa
14999992 correct: bbbbaaabaab with regex: bUaUa
14999993 correct: bab with regex: bUaUa
14999994 correct: bbaaab with regex: bUaUa
14999995 correct: abaaaa with regex: bUaUa
14999996 correct: abaaabbaaa with regex: bUaUa
14999997 correct: babbbabbbabb with regex: bUaUa
14999998 correct: aaaabbb with regex: bUaUa
14999999 correct: bbbb with regex: bUaUa
15000000 correct: aabaab with regex: bUaUa
15000001 correct: abababaabaaa with regex: bUaUa
15000002 correct: baaa with regex: bUaUa
15000003 correct: aababbaabaaa with regex: bUaUa
15000004 correct: aabbaabbbabb with regex: bUaUa
15000005 correct: bbbba with regex: bUaUa
15000006 correct: abbbbaabab with regex: bUaUa
15000007 correct: abbbbabbbbaa with regex: bUaUa
15000008 correct: ababababaaab with regex: bUaUa
15000009 correct: bb with regex: bUaUa
15000010 correct: aabaabbaa with regex: bUaUa
15000011 correct: aaaabbbababb with regex: ea
15000012 correct: abaabaababaab with regex: ea
15000013 correct: aabbbbaababb with regex: ea
15000014 correct: aab with regex: ea
15000015 correct: bbabbaab with regex: ea
15000016 correct: b with regex: ea
15000000,1 58%
```

# Questions?