NFA2ParikhImage

Caleb Cheng
2018 Summer Intern
Academia Sinica
Institute of Information Science

Nondeterministic Finite Automaton

RegEx: (a|b)* a

Accepts:

a

ba

aaa

aba

...

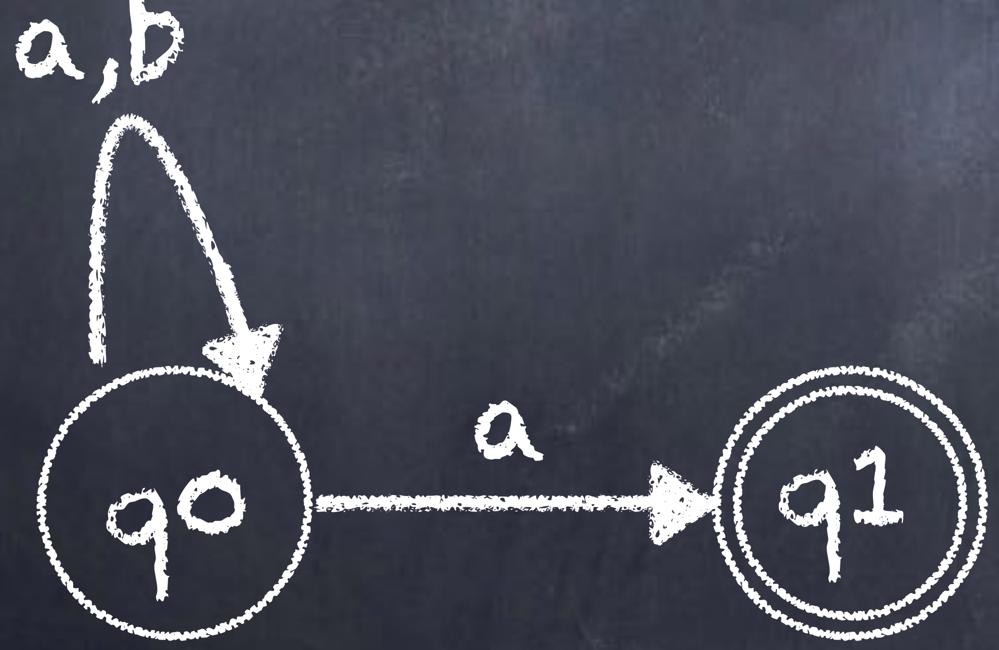
Does not Accepts:

b

bb

aab

...



Parikh Image

- The relation between the number of occurrences of each symbol

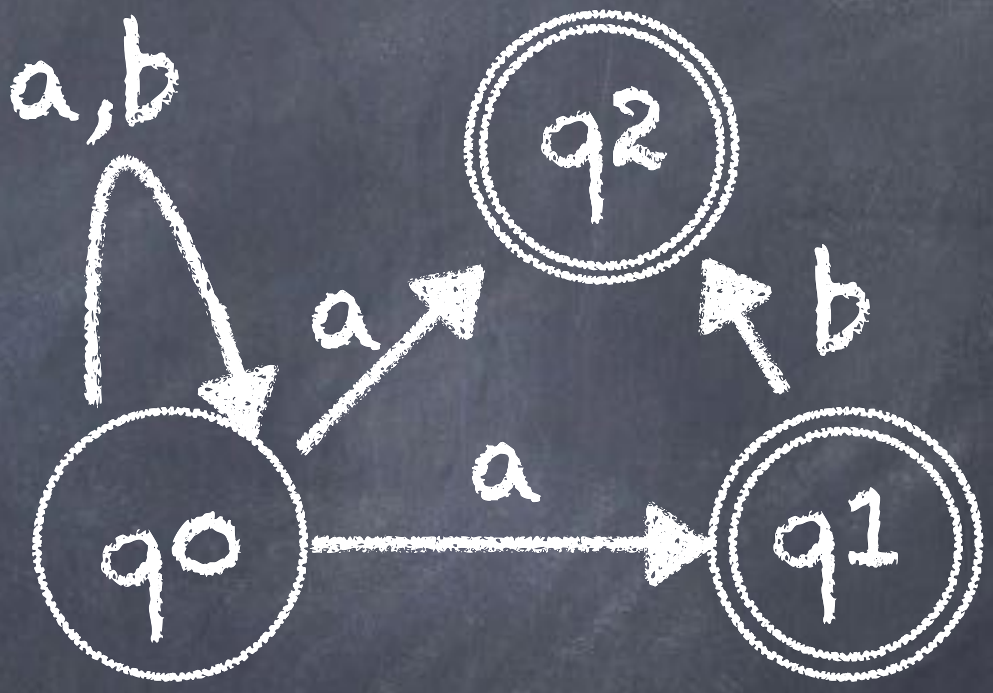
- useful for deciding whether or not a string with a given number of some terminals is accepted by a context-free grammar

Step

1. set variables (x, in, out, r...)
2. set flags
3. check connectivity
4. produce z3 constraints
5. test result using z3

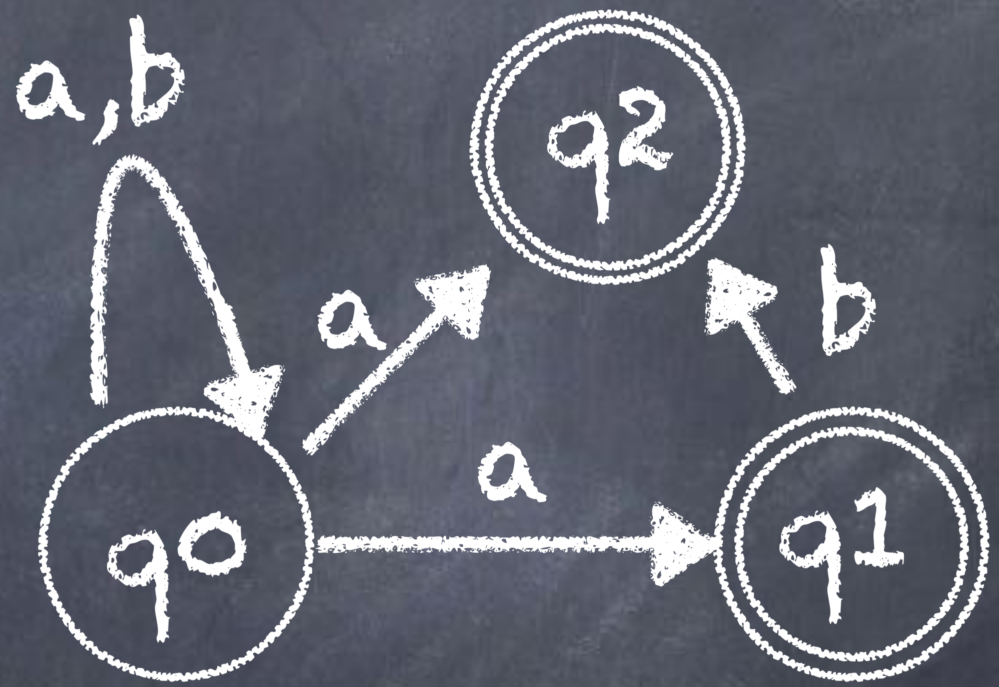
Variables and Parameters

- start_q0 (Bool)
- t_q0 (Bool)
- flag_q0_1 (Bool)
- flag_q0_2 (Bool)
- flag_q0_3 (Bool)



Variables and Parameters

- r_a (Int)
- x_q0_a_q1 (Int)
- in_q0 (Int)
- out_q0 (Int)

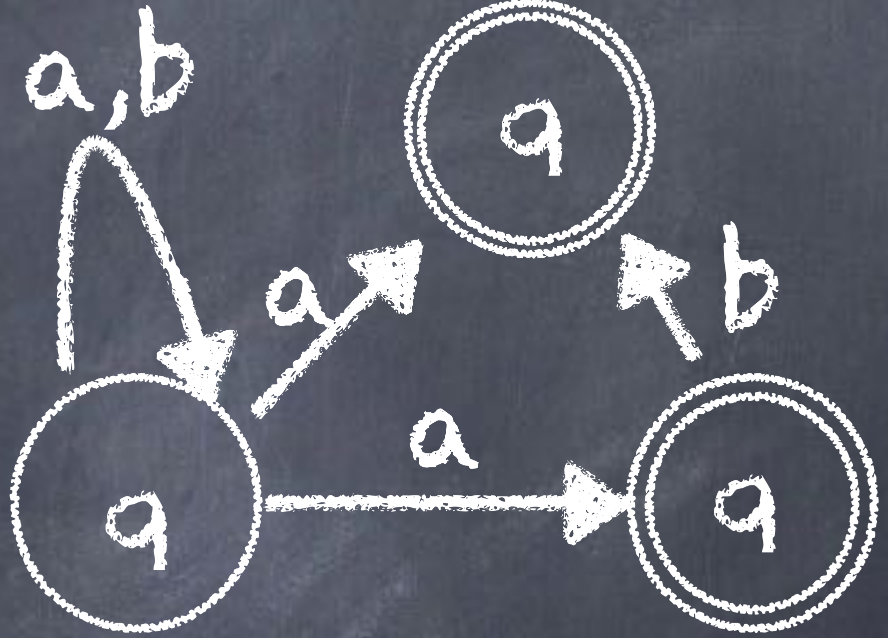


Step

1. set variables (x, in, out, r...)
2. set flags
3. check connectivity
4. produce z3 constraints
5. test result using z3

Flags (pseudocode)

```
for q in states:
    if q is terminal state and not initial state:
        flag_q_1 = true
        in_q = out_q + 1
    else if q is initial state and not not terminal state:
        flag_q_2 = true
        in_q = out_q - 1
    else
        flag_q_3 = true
        in_q = out_q
```



Step

1. set variables (x, in, out, r...)
2. set flags
3. check connectivity
4. produce z3 constraints
5. test result using z3

Check connectivity

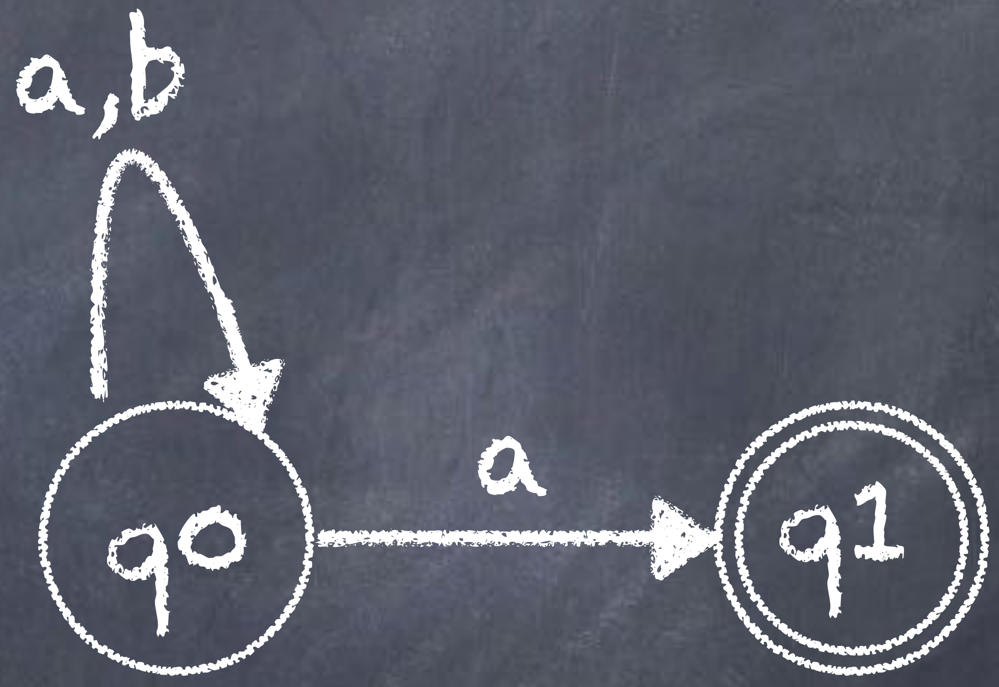
- Depth First Search
- if not reachable from the initial state, set out to 0

Step

1. set variables (x, in, out, r...)
2. set flags
3. check connectivity
4. produce z3 constraints
5. test result using z3

Output

```
(assert (and (and (and (and (and (and
(and (and (and (and (and (and (and (and
(and (and (and (= r_a (+ x_q0_a_q0
x_q0_a_q1 ))(= r_b x_q0_b_q0 ))(= out_q0
(+ x_q0_a_q0 (+ x_q0_b_q0 x_q0_a_q1 ))))
(= out_q1 0 ))(= in_q0 (+ x_q0_a_q0
x_q0_b_q0 )))(= in_q1 x_q0_a_q1 ))start_q0
)(not  start_q1 ))(= t_q0 0 ))(= 1 t_q1 ))(or
(or flag_q0_1 flag_q0_2 )flag_q0_3 ))(=
flag_q0_1 (and (and (not start_q0  )(= t_q0
1 ))(= in_q0 (+ out_q0 1 )))))(= flag_q0_2
(and (and start_q0 (= t_q0 0 ))(= in_q0 (-
out_q0 1 )))))(= flag_q0_3 (and (= in_q0
out_q0 )(not (xor start_q0 (= t_q0 1 )) ))))
(or (or flag_q1_1 flag_q1_2 )flag_q1_3 ))(=
flag_q1_1 (and (and (not start_q1  )(= t_q1
1 ))(= in_q1 (+ out_q1 1 )))))(= flag_q1_2
(and (and start_q1 (= t_q1 0 ))(= in_q1 (-
out_q1 1 )))))(= flag_q1_3 (and (= in_q1
out_q1 )(not (xor start_q1 (= t_q1 1 )) )))))
```



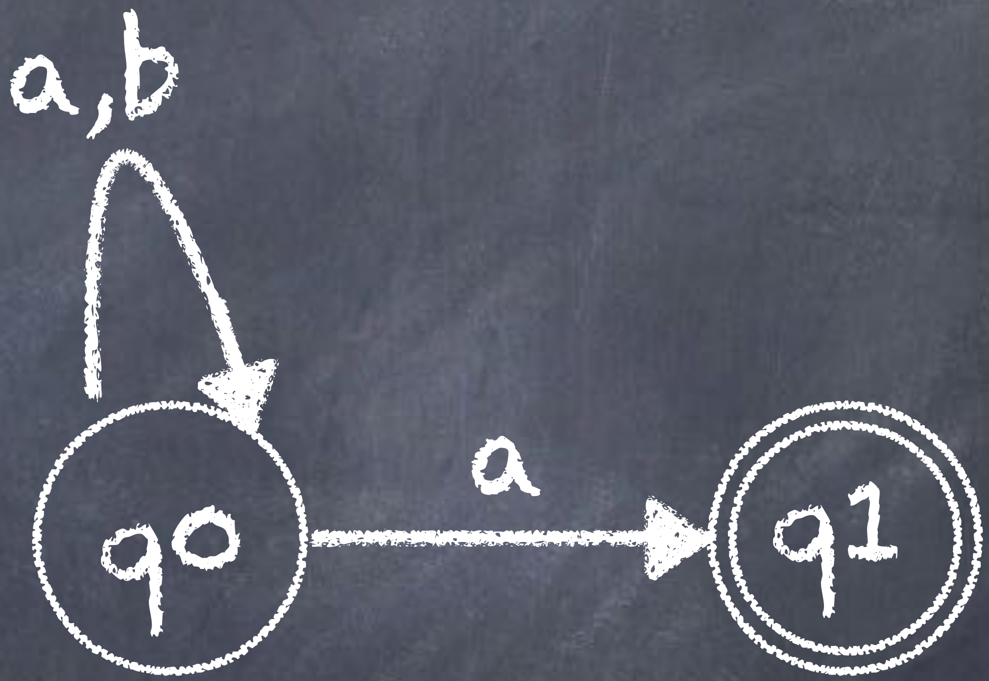
Step

1. set variables (x, in, out, r...)
2. set flags
3. check connectivity
4. produce z3 constraints
5. test result using z3

Test by z3

SAT

- (define-fun x_q0_b_q0 () Int 0)
- (define-fun x_q0_a_q0 () Int 0)
- (define-fun flag_q1_3 () Bool false)
- (define-fun flag_q1_2 () Bool false)
- (define-fun in_q1 () Int 1)
- (define-fun flag_q1_1 () Bool true)
- (define-fun flag_q0_3 () Bool false)
- (define-fun flag_q0_1 () Bool false)
- (define-fun flag_q0_2 () Bool true)
- (define-fun t_q1 () Int 1)
- (define-fun t_q0 () Int 0)
- (define-fun start_q1 () Bool false)
- (define-fun start_q0 () Bool true)
- (define-fun x_q0_a_q1 () Int 1)
- (define-fun in_q0 () Int 0)
- (define-fun out_q1 () Int 0)
- (define-fun out_q0 () Int 1)
- (define-fun r_b () Int 0)
- (define-fun r_a () In 1)



Reference

- https://hal.archives-ouvertes.fr/hal-00159525/document

Code

- https://github.com/CodingSheep1229/NFA2ParikhImageCI

Q & A

THE END

thank you