1

Regular Expressions

An active presentation by

Pascal Führlich, Kim Borchart und Corinna Jaschek

Programming Experience - SS18 - Prof. Dr. Robert Hirschfeld, Jens Lincke, Stefan Ramson

"What do you call a phobia of regexes?"

"What do you call a phobia of regexes?"

"Common Sense!"

Regex Definition

- **in theory:** defining a regular language
- **in practice:** finding out which (sub)strings match a pattern
- **implementation approaches:** backtracking, NFA, DFA

What are regexes used for?

- **data validation** (is this a correct email?)
- **data extraction** (get all emails out of a text)
- **search and replace**

Enter a regex that matches all of these:

ab

b

and none of these entries:

aab

abbb

Explain Regex

8

'aaa' matchesRegex: 'a?a?a?aaa'

'aaa' matchesRegex: 'a?a?a?aaa'

aaa

Start Stepping

'aaaaaa' matchesRegex: 'a?a?a?a?aaaa'

aaaaaa

Start Stepping

'aaaa' matchesRegex: 'a?a?a?a?aaaa'

aaaa

Start Stepping

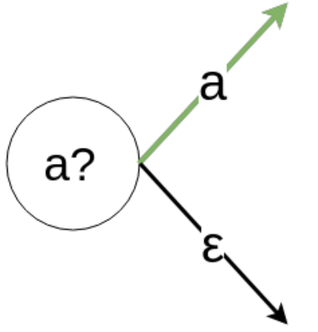
'aaaaaaaa' matchesRegex: 'a?a?a?a?aaaa'

aaaaaaaa

Start Stepping

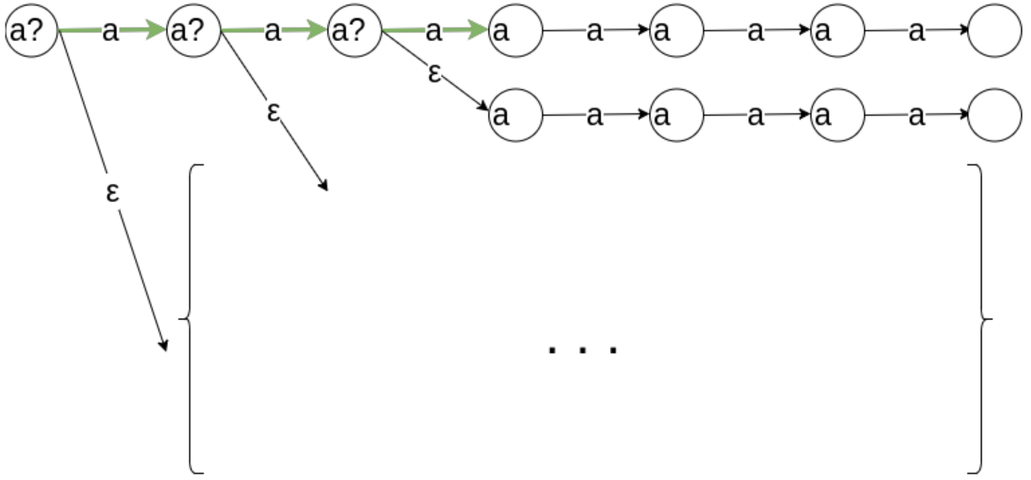
Regex As Decisions:

Decision: a?



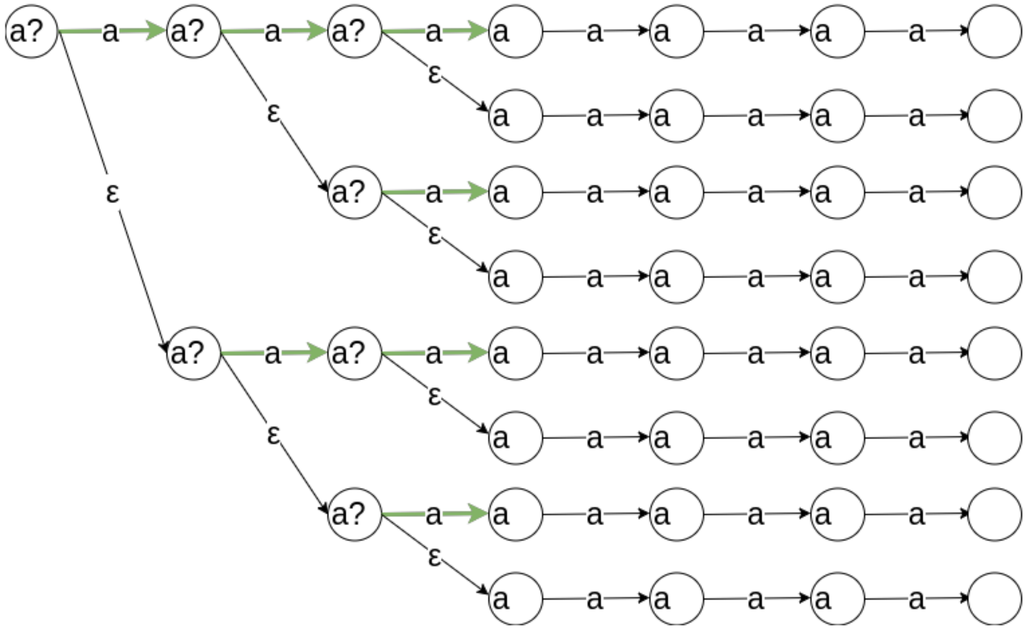
Regex As Decisions:

Example: 'aaa' matchesRegex: 'a?a?a?aaa'



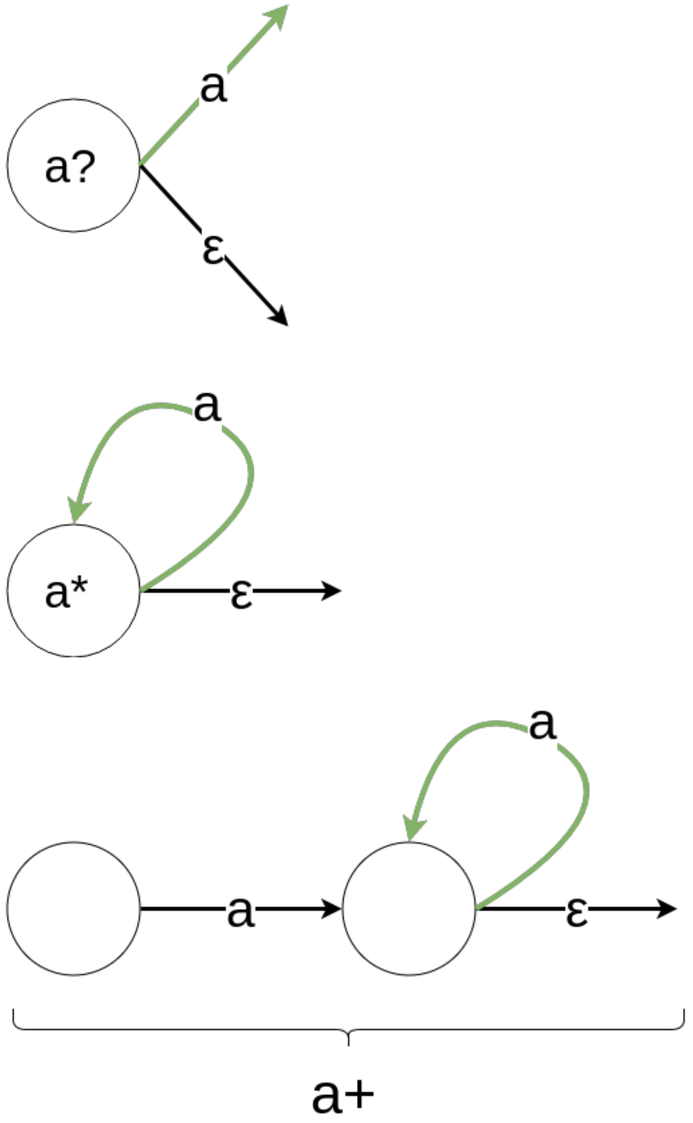
Regex As Decisions:

Example: 'aaa' matchesRegex: 'a?a?a?aaa'



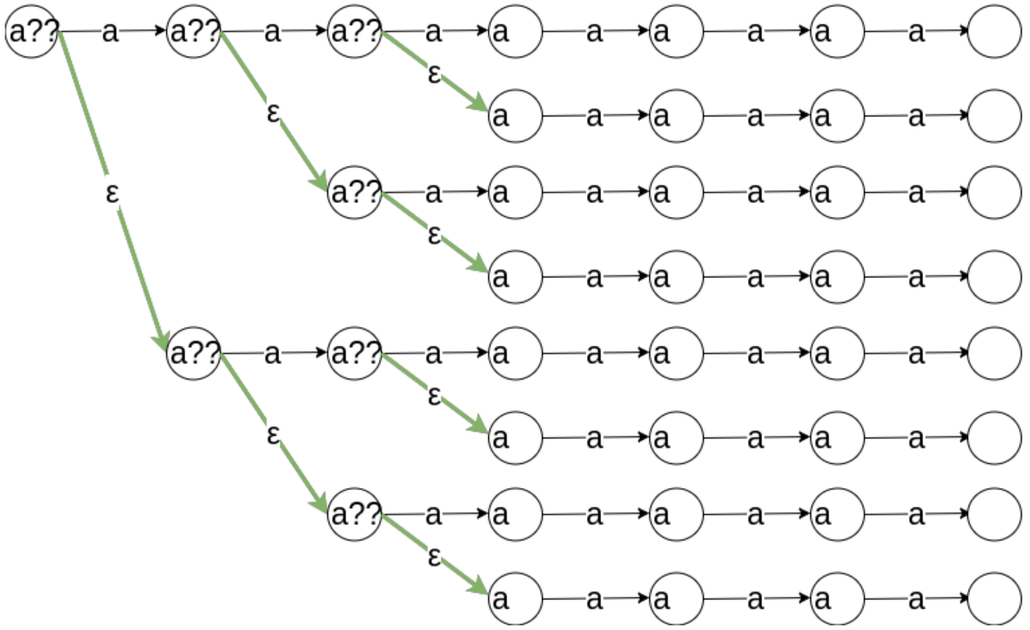
Regex As Decisions:

Decision: a?, a*, a+



Regex As Decisions:

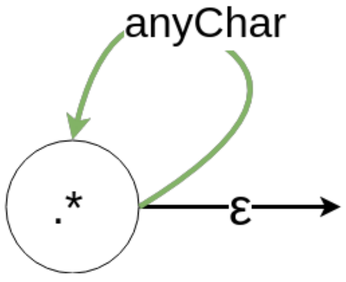
Example: 'aaa' matchesRegex: 'a??a??a??aaa'



'Tobias'

'Tobias'

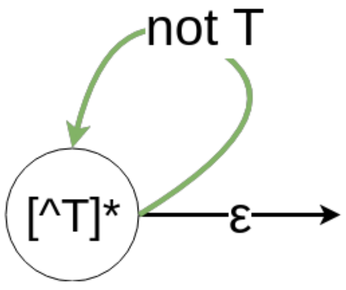
'.*Tobias'



'Tobias' matchesRegex: '.*Tobias'

Tobias

Start Stepping

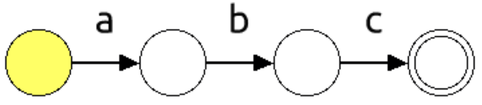


'Tobias' matchesRegex: '[^T]*Tobias'

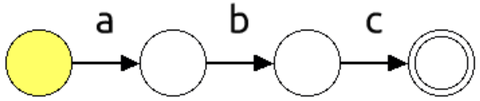
Tobias

Start Stepping

DFA - Deterministic Finite Automaton

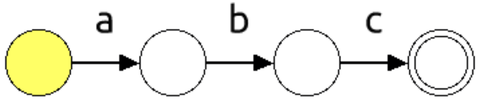


DFA - Deterministic Finite Automaton

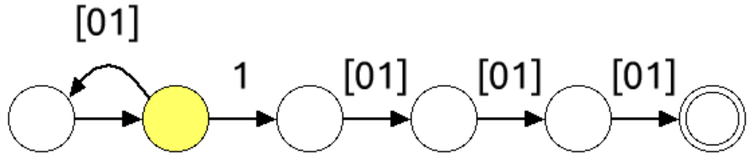


- O(n) runtime, where n = |string|
- O(2^m) states/construction, where m = |regex|

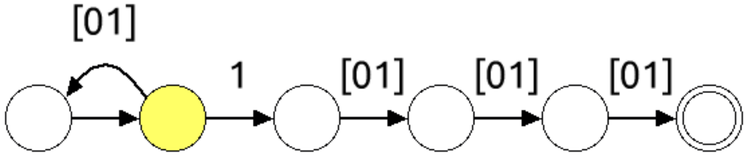
NFA - Nondeterministic Finite Automaton



NFA - Nondeterministic Finite Automaton

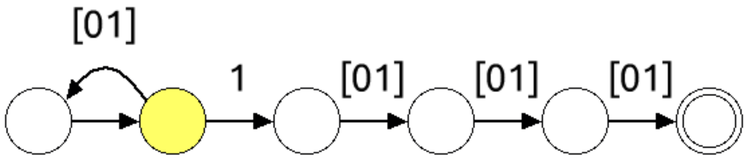


NFA - Nondeterministic Finite Automaton



[01]*1[01][01][01]

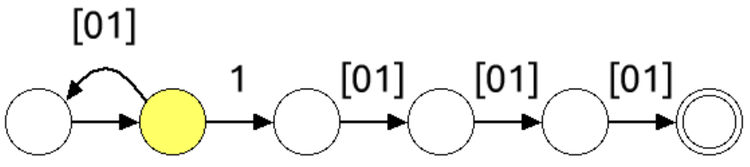
NFA - Nondeterministic Finite Automaton



[01]*1[01][01][01]

- in theory: guess correct path
- in practice: try all in parallel

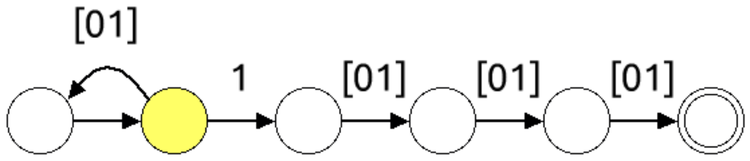
NFA - Simulation



[01]*1[01][01][01]

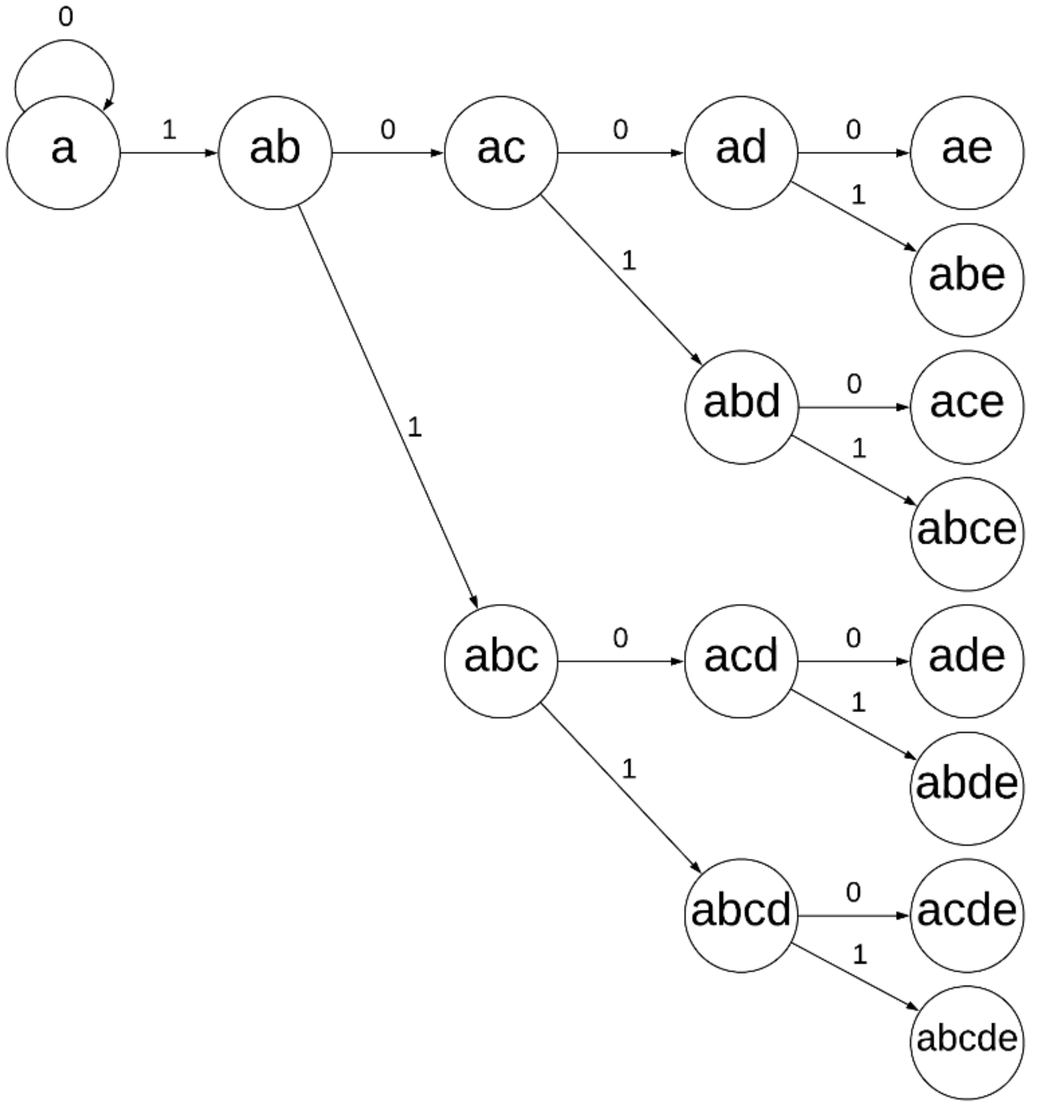
Enter search text here...

NFA to DFA - Powerset Construction

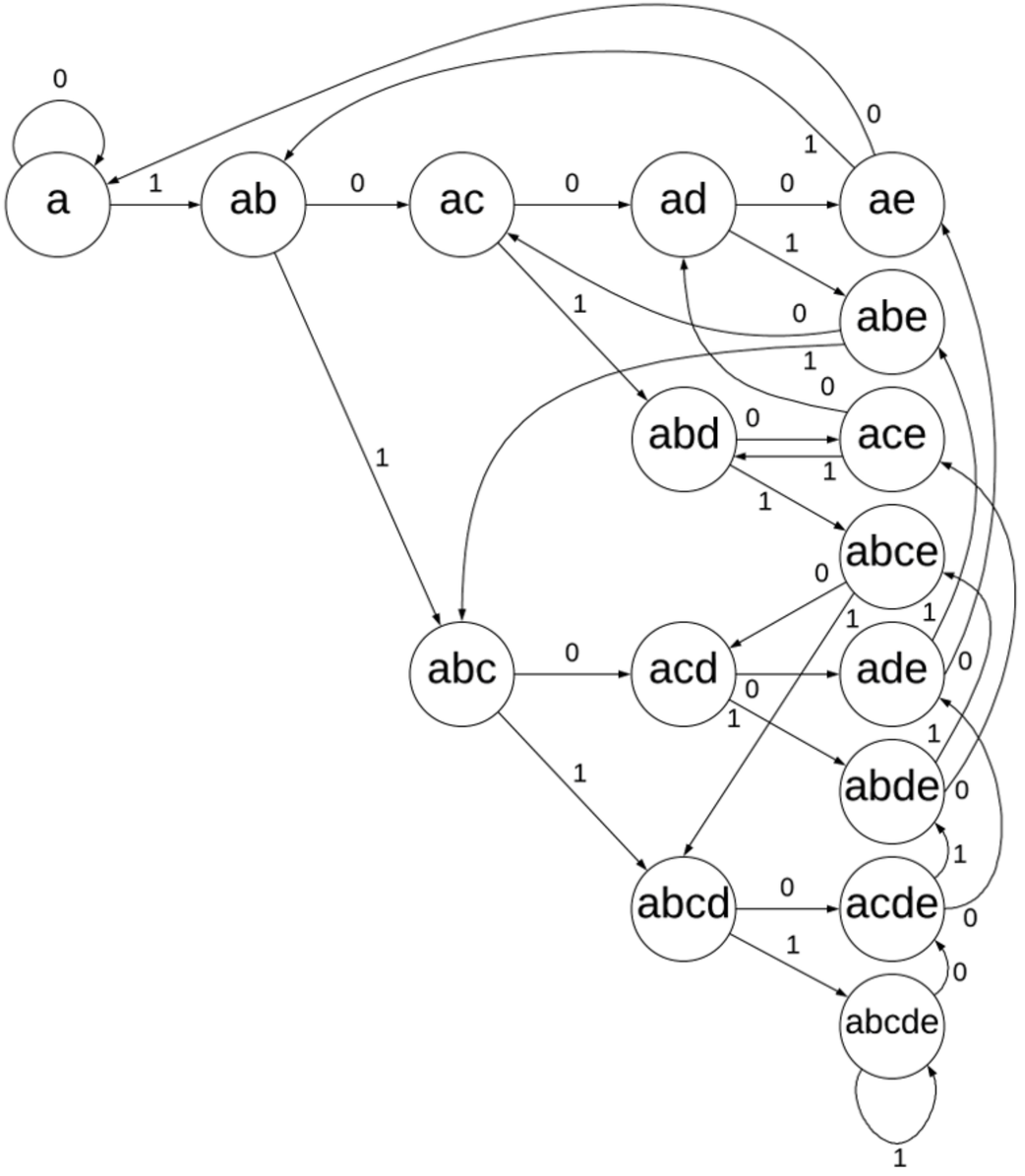


[01]*1[01][01][01]

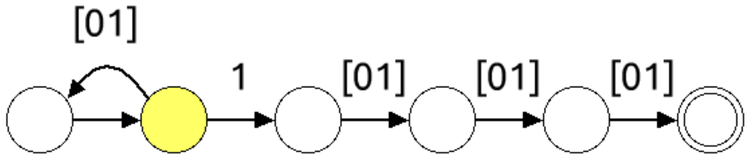
Enter search text here...



NFA to DFA - Powerset Construction



NFA - Complexity



- easy to read
- O(m) states/construction, where m = |regex|
- O(m * n) runtime, where n = |string|

NFA Construction - Thompsons Algorithm

Regex

a(bc|d)

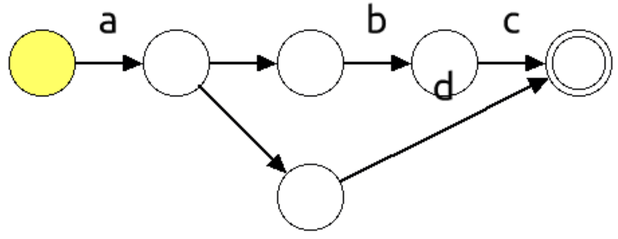
↓

Postfix Regex

abc.d|.

↓

NFA



NFA Construction - Thompsons Algorithm

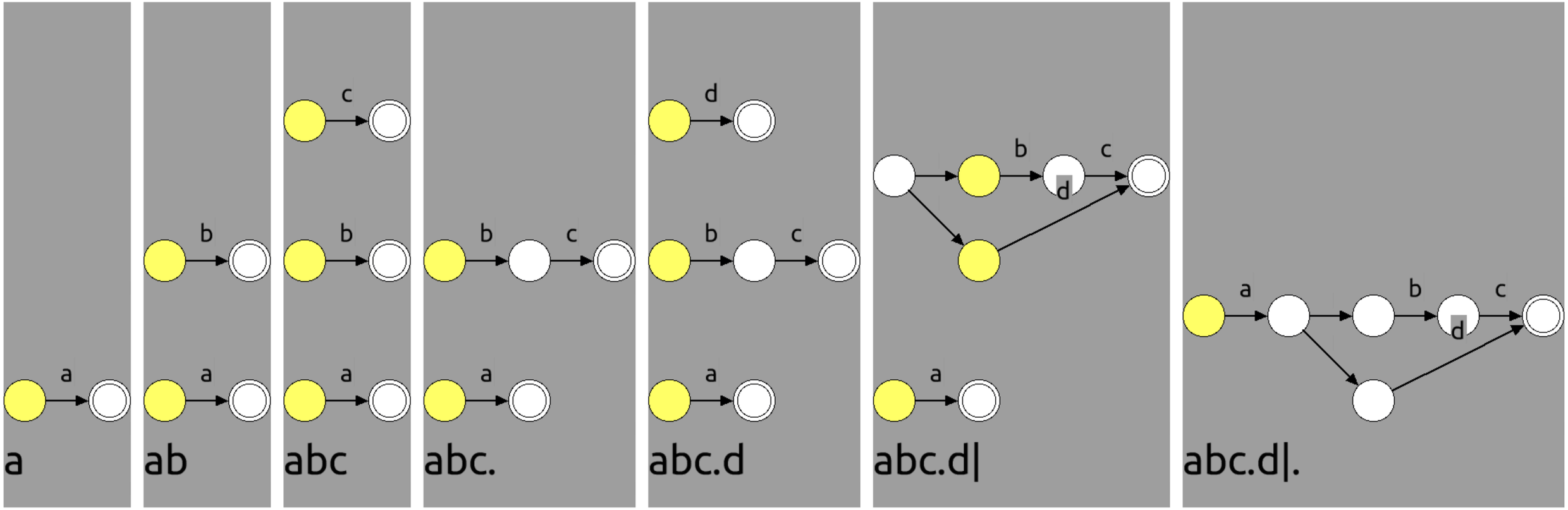
Regex --> Postfix Regex

Massive switch case...

NFA Construction - Thompsons Algorithm

Regex --> Postfix Regex --> NFA

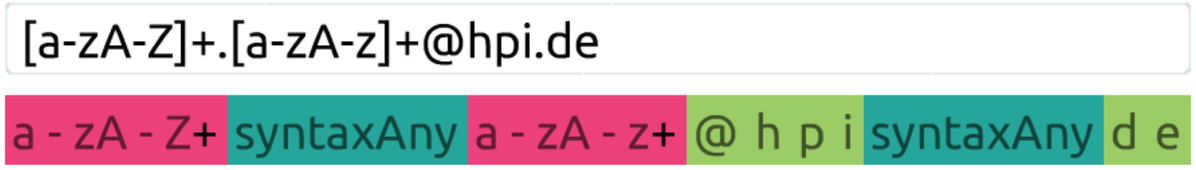
a(bc|d) --> abc.d|.



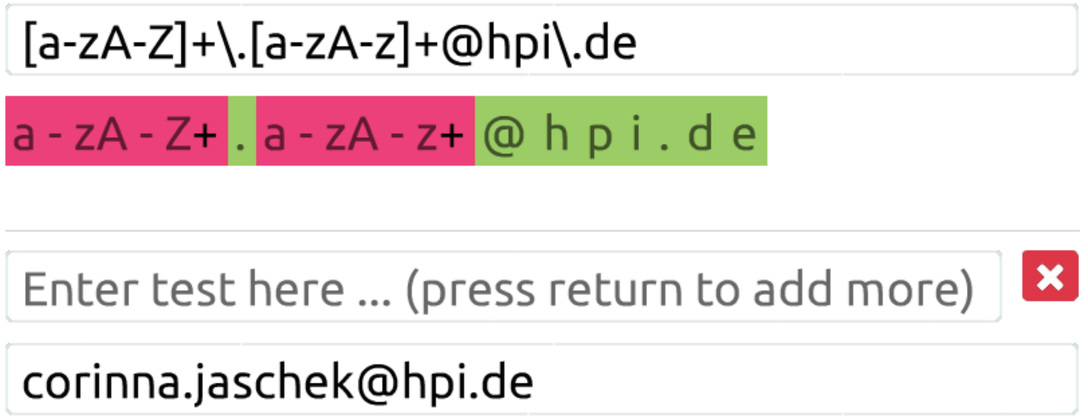
Recap

- Most modern regex engines use backtracking
- Backtracking is powerful, but can be VERY slow
- NFAs are less powerful, but ALWAYS efficient

Syntax Explanation



Test Cases

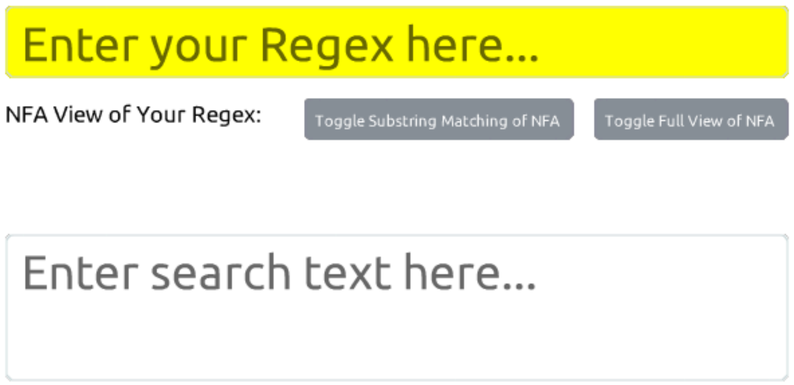

[a-zA-Z]+\.[a-zA-z]+@hpi\.de
a - zA - Z+ . a - zA - z+ @ h p i . d e
Enter test here ... (press return to add more)
corinna.jaschek@hpi.de

Substring Matching

```
[a-z.0-9]+@[a-z]+(\.[a-z]+)+
```

```
This is my text.
It contains several email addresses like jane.doe@hpi.de and john@smith.com.
Try matching them all: abc123@test.co.uk and anothertest@x.yz.
But don't match something@test.
```

Dynamic NFA


Enter your Regex here...
NFA View of Your Regex:
Toggle Substring Matching of NFA
Toggle Full View of NFA
Enter search text here...

Quick Debugging