

# Automata and Formal Languages (6)

Email: christian.urban at kcl.ac.uk  
Office: S1.27 (1st floor Strand Building)  
Slides: KEATS (also home work is there)

"I hate coding. I do not want to look at code."

"I am appalled. You do not show code anymore."

# ReDoS

- Regular **e**xpression **D**enial of **S**ervice
- "Regular Expressions Will Stab You in the Back"
- Evil regular expressions
  - $(a?\{n\})a\{n\}$
  - $(a^+)^+$
  - $([a - zA - Z]^+)^*$
  - $(a + aa)^+$
  - $(a + a?)^+$

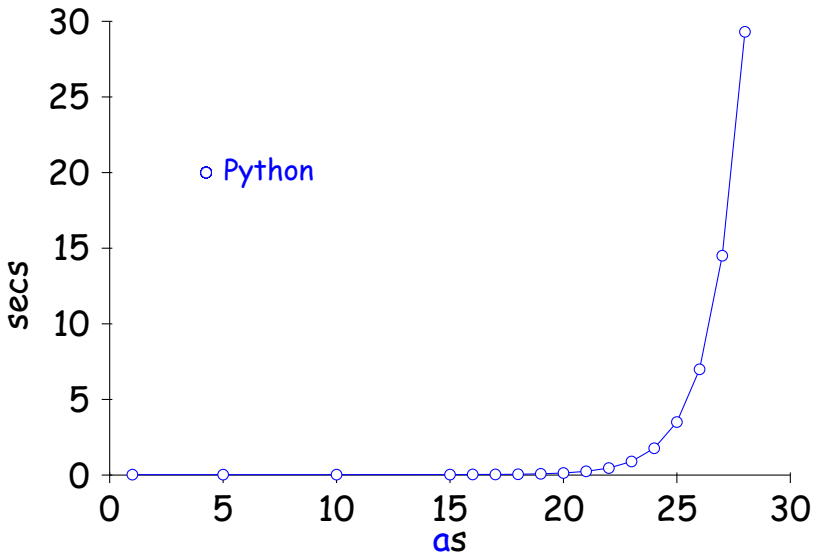
# Regex Matching

Given a regular expression

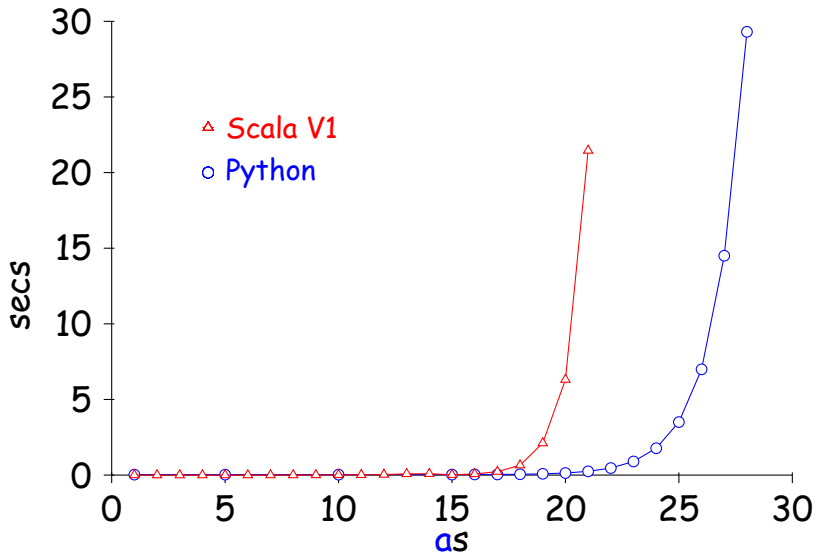
- 1 you might convert it into a DFA (subset construction)
- 2 you might try all possible paths in an NFA via backtracking
- 3 you might try all paths in an NFA in parallel
- 4 you might try to convert the DFA "lazily"

Often No 2 is implemented (sometimes there are even good reasons for doing this).

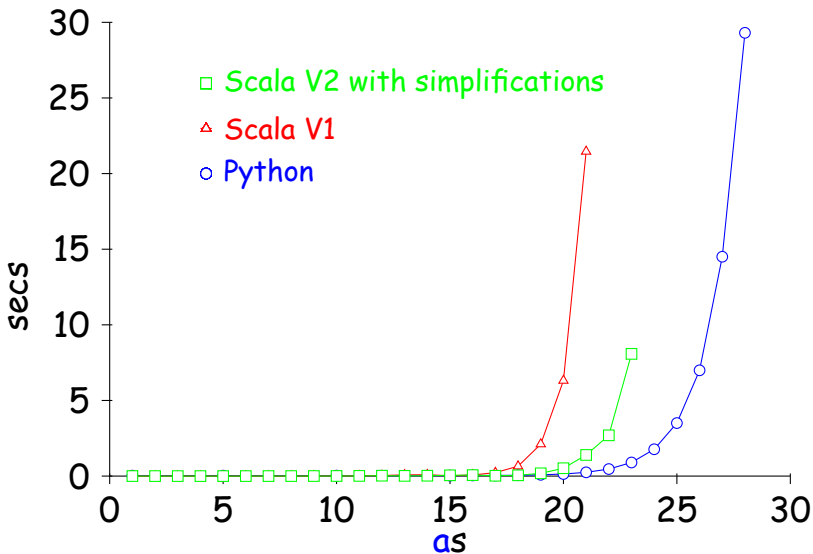
# $(a?\{n\})a\{n\}$ in Python



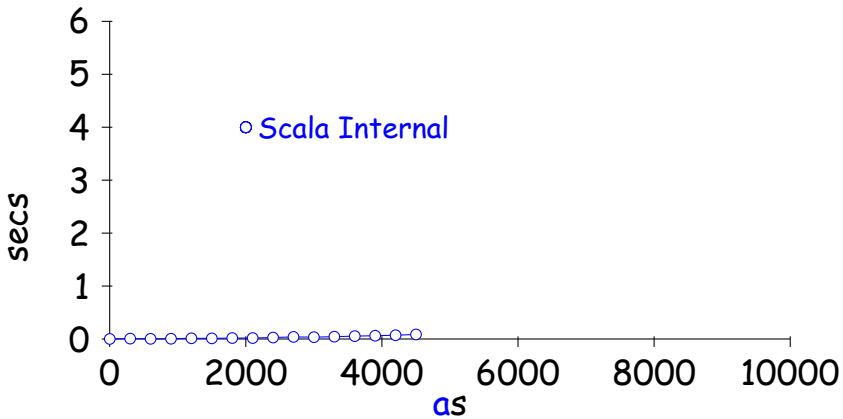
# $(a?\{n\})a\{n\}$ in Python

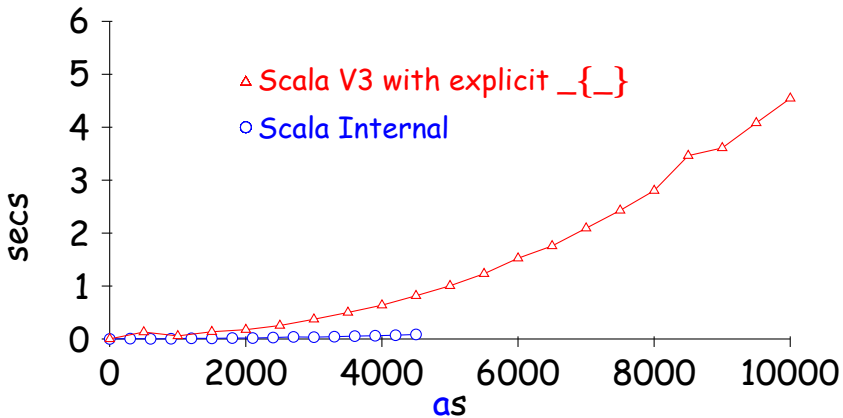


# $(a?\{n\})a\{n\}$ in Python









# Grammars

A (context-free) Grammar  $G$  consists of

- a finite set of nonterminal symbols (upper case)
- a finite terminal symbols or tokens (lower case)
- a start symbol (which must be a nonterminal)
- a set of rules

$$A \rightarrow rhs$$

where  $rhs$  are sequences involving terminals and nonterminals.

# Grammars

A (context-free) Grammar  $G$  consists of

- a finite set of nonterminal symbols (upper case)
- a finite terminal symbols or tokens (lower case)
- a start symbol (which must be a nonterminal)
- a set of rules

$$A \rightarrow \text{rhs}$$

where  $\text{rhs}$  are sequences involving terminals and nonterminals.

We can also allow rules

$$A \rightarrow \text{rhs}_1 | \text{rhs}_2 | \dots$$

# Palindromes

$$S \rightarrow \epsilon$$

$$S \rightarrow aSa$$

$$S \rightarrow bSb$$

# Palindromes

$$S \rightarrow \epsilon$$

$$S \rightarrow aSa$$

$$S \rightarrow bSb$$

or

$$S \rightarrow \epsilon \mid aSa \mid bSb$$

# Arithmetic Expressions

$E \rightarrow \text{num\_token}$

$E \rightarrow E + E$

$E \rightarrow E - E$

$E \rightarrow E * E$

$E \rightarrow (E)$

# Arithmetic Expressions

$E \rightarrow num\_token$

$E \rightarrow E + E$

$E \rightarrow E - E$

$E \rightarrow E * E$

$E \rightarrow (E)$

1 + 2 \* 3 + 4



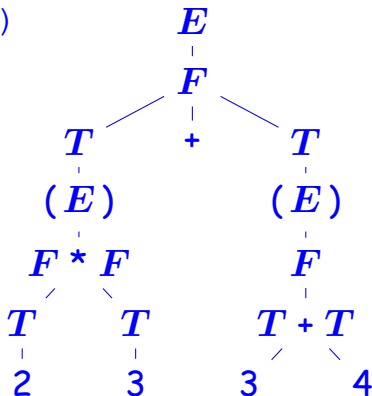
# Parse Trees

$$E \rightarrow F \mid F * F$$

$$F \rightarrow T \mid T + T \mid T - T$$

$$T \rightarrow \text{num\_token} \mid (E)$$

$(2 * 3) + (3 + 4)$



# Ambiguous Grammars

A grammar is **ambiguous** if there is a string that has at least two parse trees.

$$E \rightarrow \textit{num\_token}$$

$$E \rightarrow E + E$$

$$E \rightarrow E - E$$

$$E \rightarrow E * E$$

$$E \rightarrow (E)$$

1 + 2 \* 3 + 4

# Chomsky Normal Form

All rules must be of the form

$$A \rightarrow a$$

or

$$A \rightarrow BC$$

# CYK Algorithm