WORKSHEET

A6.4 Regular expressions and regular languages

Theory of computation · A level · AQA 7517 4.4.2.3 · about 25 min

Name _____ Class _____ Date _____

What this lesson is about

The metacharacters, matching in Python, the link between regexes and FSMs, and what makes a language regular.

- Languages
- The metacharacters
- Regular expressions in Python
- Regular expressions and FSMs
- Regular languages

Questions

6 marks in all

1. Which strings match the regular expression ab*a? [1 mark]

Tick every answer that is true.

- ☐ **A** aa
- ☐ **B** aba
- ☐ **C** abba
- ☐ **D** ab
- ☐ **E** baa

2. What does the regular expression (a|b)+ describe? [1 mark]

- ☐ **A** Every string of a and b, including the empty string
- ☐ **B** Every non-empty string of a and b
- ☐ **C** Exactly one a or one b
- ☐ **D** The string ab repeated

3. Which statement about regular expressions and finite state machines is true? [1 mark]

- ☐ **A** Every regex has an FSM that accepts the same language
- ☐ **B** FSMs can describe more languages than regexes
- ☐ **C** Regexes can describe more languages than FSMs
- ☐ **D** Only regexes can describe infinite languages

4. Why is the language $\{0^n1^n \mid n \geq 1\}$ not regular? [1 mark]

- ☐ **A** It is infinite
- ☐ **B** An FSM would need to count an unlimited number of 0s with a finite number of states
- ☐ **C** It uses two symbols
- ☐ **D** It contains the empty string

5. Write a regular expression, using only 0, 1 and the metacharacters * + ? | ( ), for binary strings that start with 1 and end with 0. [1 mark]

6. What does this program print? [1 mark]

```
import re
for s in ["ac", "abbc", "abcc", "c"]:
    print(s, re.fullmatch(r"ab*c", s) is not None)
```

The task: three languages

Write a regular expression for each language, then test it. - `L1` : strings over `{a, b}` that start with `a` and end with `b` . - `L2` : binary strings in which every `1` is immediately followed by a `0` . The string `000` counts, because it has no 1s at all. - `L3` : binary numbers with no leading zeros: `0` itself, or a `1` followed by any binary digits. Rules: - Assign each pattern to its variable as a raw string on one line, such as `L1 = r"..."` . Use only the symbols of the language and the metacharacters `* + ? | ( )` . No `[ ]` , no `\d` . - Test with `re.fullmatch` . For every string in `tests[name]` , working through `L1` , `L2` , `L3` in that order, print the name, the string and `yes` or `no` , separated by single spaces: for example `L1 ab yes` . Fifteen lines in all.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

import re

tests = {
    "L1": ["ab", "aab", "ba", "a", "abab"],
    "L2": ["1010", "0100", "110", "01", "000"],
    "L3": ["0", "1010", "0101", "100", "00"],
}
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/a6-4-regular-expressions/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Write a regex for binary strings that contain `11` somewhere. Then draw its FSM.
2. Does `(a|b)*` include the empty string? Does `(a|b)+`?
3. Explain why no regex can describe `{aⁿbⁿ | n ≥ 1}`, but `a+b+` is fine.