Vivian Do
CSC 4330
Nov 01 2023

Homework 6

1. (25 points).

The language composed of all strings over the alphabet {a,b} of the form: zero or more a's followed by zero or more b's, i.e., {"", a, b, aa, ab, bb, aaa, aab, abb, bbb, ...}, where "" is the empty string, is a regular language: show why this is. Note that you may use any line of argument from the slides on regular languages, e.g., by (a) showing that it can be constructed using the properties on slide 5 ("Regular Language: a formal definition"), like in the examples on the subsequent slides 6 and 7, or by (b) drawing the finite state automaton (FSA), or giving its description, like in the examples on slides 13 and 14. If you draw the FSA, you may draw in on a piece of paper, take a picture, and upload that as part of the assignment.

The collection {"", a, b, aa, ab, bb, aaa, aab, abb, bbb, …} over the alphabet Σ = {a, b}

- has an empty string {""}
- includes a's & b's as singletons {a} & {b}
- includes strings of a's & b's from Kleene closure {a, aa, aaa, …} & {b, bb, bbb, …}
- includes combined strings of a's & b's from union & concatenation {ab, aab, abb, …}

Therefore, this is a regular language.

2. (25 points).

Is the language composed of all strings over the alphabet {a,b} that contain an odd number of a's a regular language? If so, show why. Note, again, that you may use any line of argument from the slides on regular languages, e.g., by (a) using the properties on slide 5, by (b) drawing the FSA, or by (c) using the additional properties on slide 9 ("Closure properties of regular languages"), along with the fact that the language on slide 7 is regular (hint, hint,..).

No, this is not a regular language.

The language composed of all strings over the alphabet Σ = {a, b} that contain an odd number

- has an empty string
- does not include a's & b's as singletons {a}
- does not include strings of a's & b's from union & concatenation

Since there is an odd number of a's in these strings, it will create an irregular pattern that does not follow the properties of regular languages.

3.

a. (10 points). What would be a regular expression (regex) for the language of exercise 1.?

b. (10 points). What would be a regex for the language of exercise 2.?

c. (10 points). Can you think of another (different, but equivalent) regex for the language of exercise 2.?

a:
(a*|b*)*

b:
b*(ab*ab*)*

- *b** allows any number of zeros or b's at the beginning of the string.
- *(ab*ab*)** matches pairs of a's with any number of b's.

c:
(b*ab*ab*)*b*

- this allows for more zeros or b’s at the beginning & end of the string.

4.

Using your favorite implementation of regular expressions (Python, grep, sed,..), input your regex from exercise 3.a, and one of your regexes from exercise 3.b (or 3.c, since they are equivalent they should generate the same result anyway), and test to see how many strings from the set of all strings over the alphabet {a,b} of length at most 5, i.e., {"", a, b, ab, .., bbbbb} are described by, that is, match, each of your input regexes above. Note: that you will have to match the *entire* string with the regex, so watch out for this implementation detail.

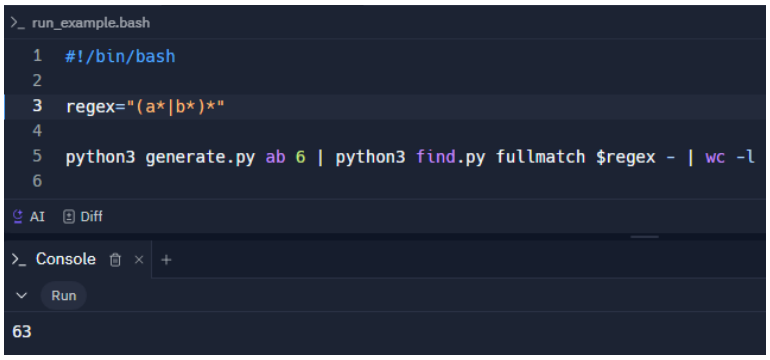
How many strings (you can just give the count) are matched with

a. (10 points). your regex from exercise 3.a?

b. (10 points). your regex from exercise 3.b (or 3.c)?

a:
63

b:
32

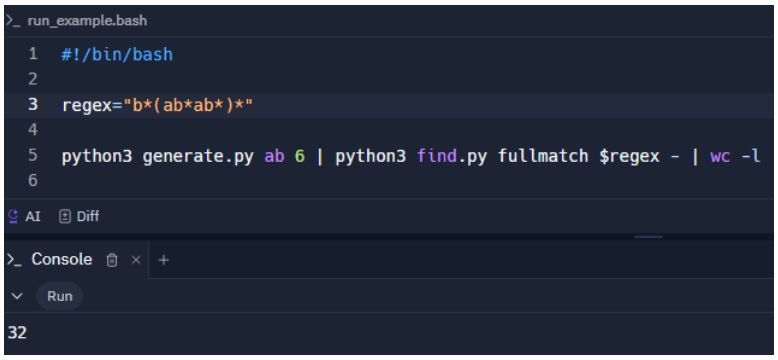


```
#!/bin/bash

regex="(a*|b*)*"

python3 generate.py ab 6 | python3 find.py fullmatch $regex - | wc -l
```

```
63
```



```
#!/bin/bash

regex="b*(ab*ab*)*"

python3 generate.py ab 6 | python3 find.py fullmatch $regex - | wc -l
```

```
32
```