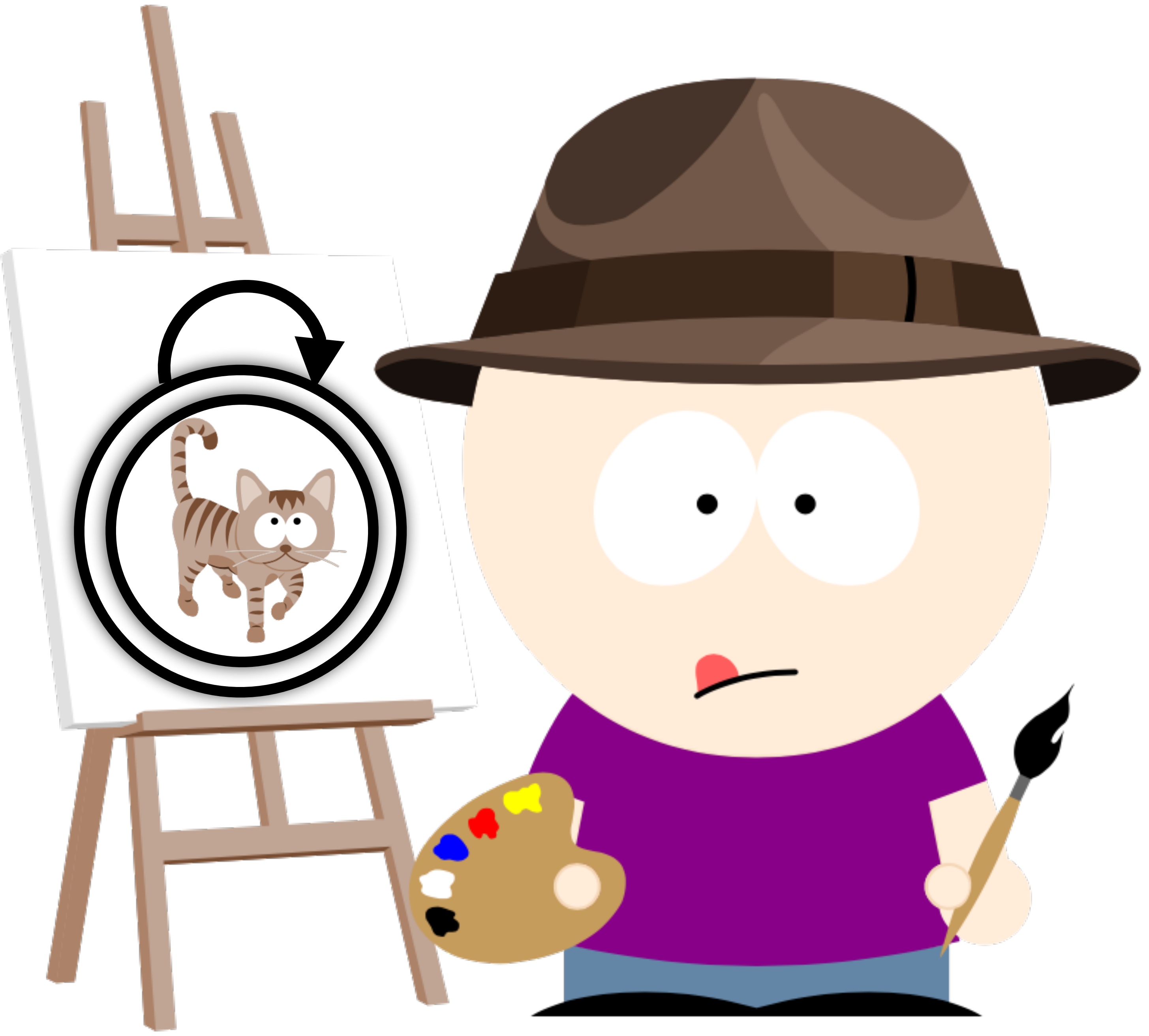
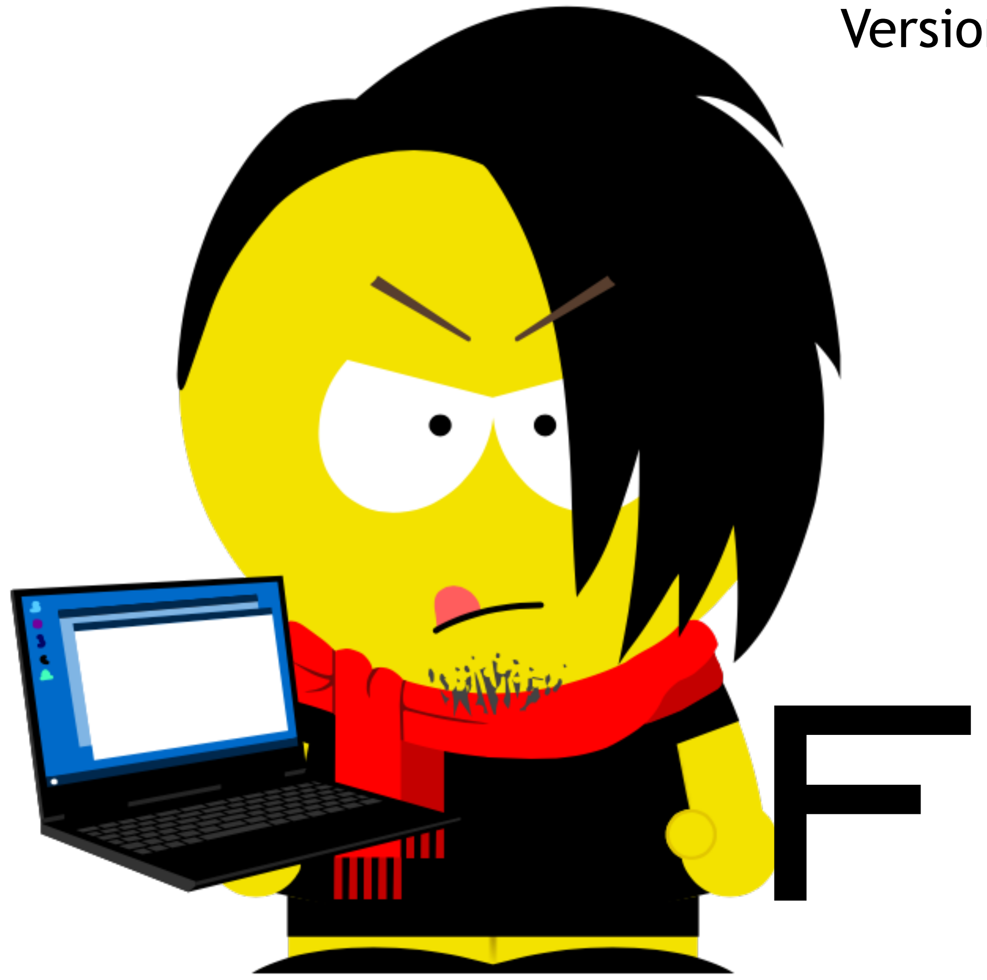


In-Class Exercises

Version: 2025-12-28



Lexical Analysis 2- Finite State Machines

Download exercises

1. Go to

   https://ligerlabs.org/compilers.html

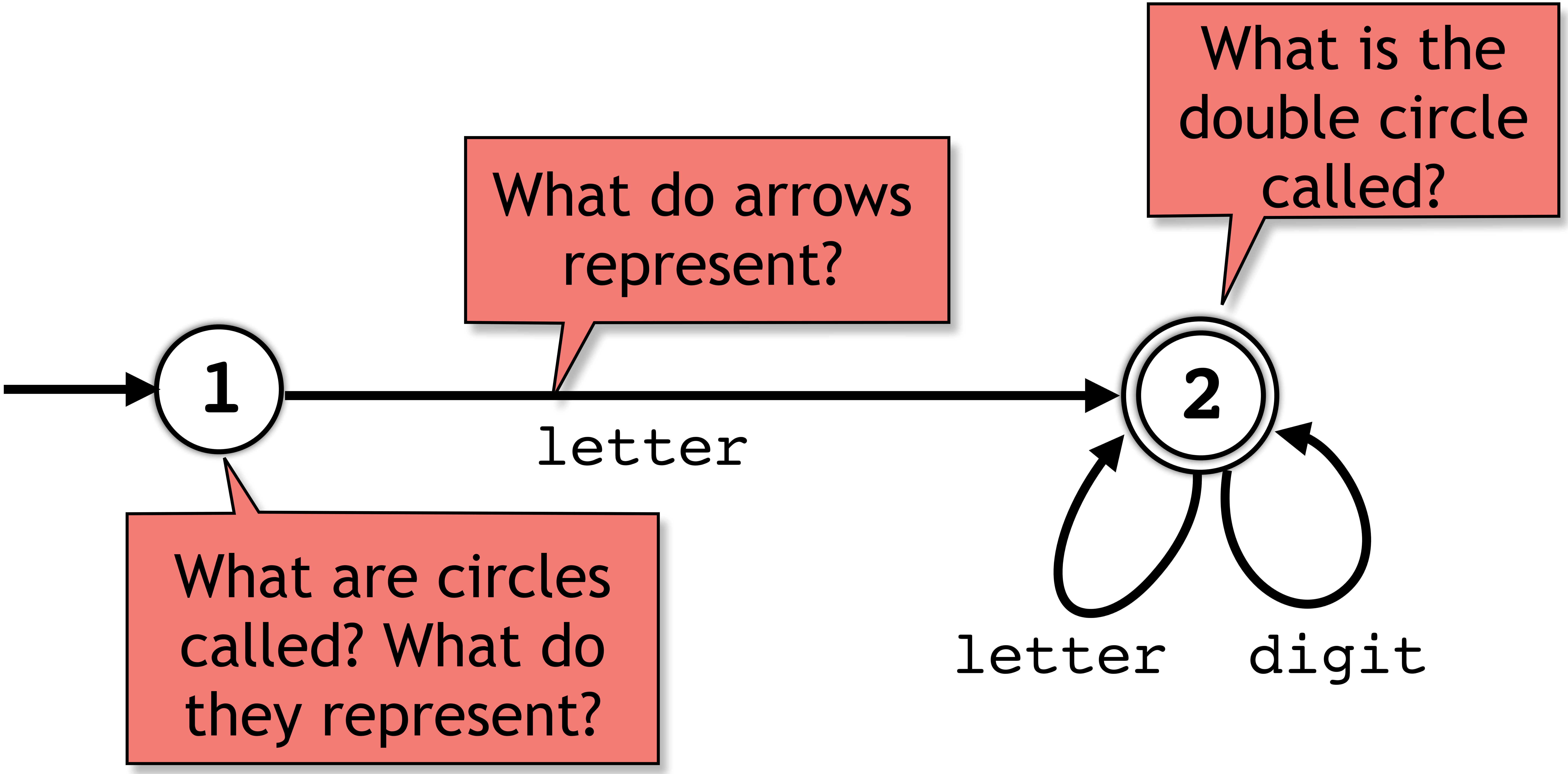
2. Download the file

   `lex-2-exercises.zip`

3. Open up a terminal and Unzip the file

```
> unzip lex-2-exercises.zip
> cd lex-2-exercises
> ls
```

Task 1

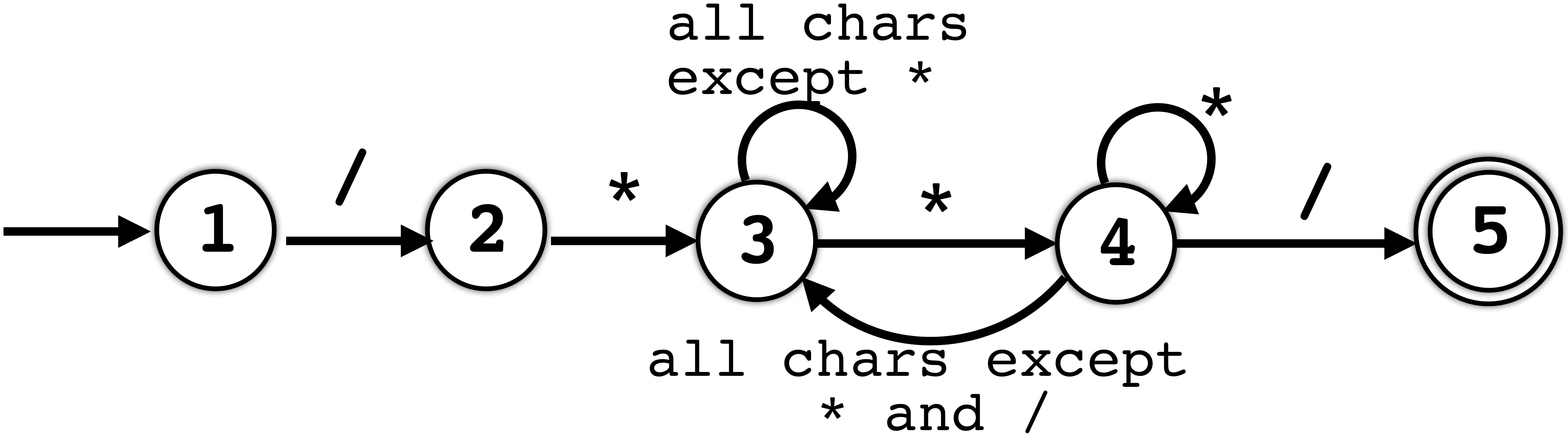


What do arrows represent?

What is the double circle called?

What are circles called? What do they represent?

Task 2



- Show the sequence of transitions that this DFA goes through when matching the string

/*6*7*/

Task 3 (a)

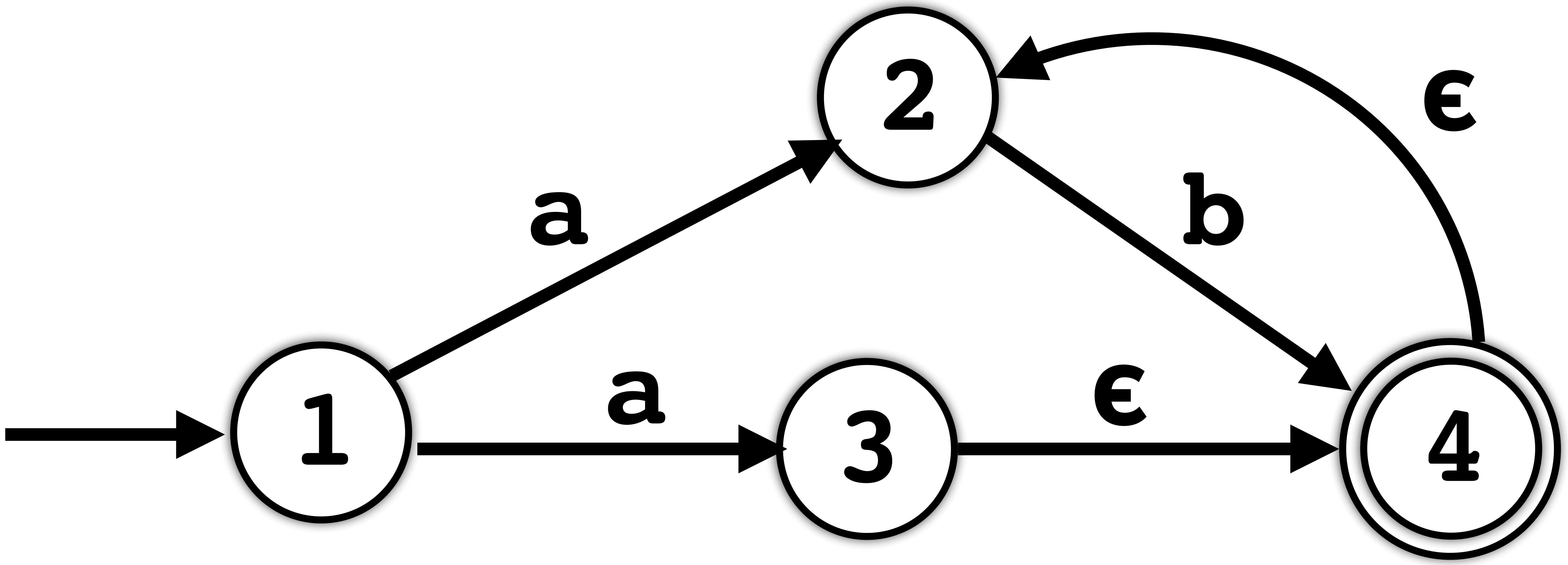
- Show the sequence of transitions that the DFA on the next page goes through when matching the strings

1. `"+67"`
2. `"42."`
3. `"42.0"`
4. `"1.0E+4"`

Task 3 (b)

digit
digit
+
1
2
digit
3
.
4
digit
5
–
E
digit
E
digit
+
6
7
digit
8
–
digit

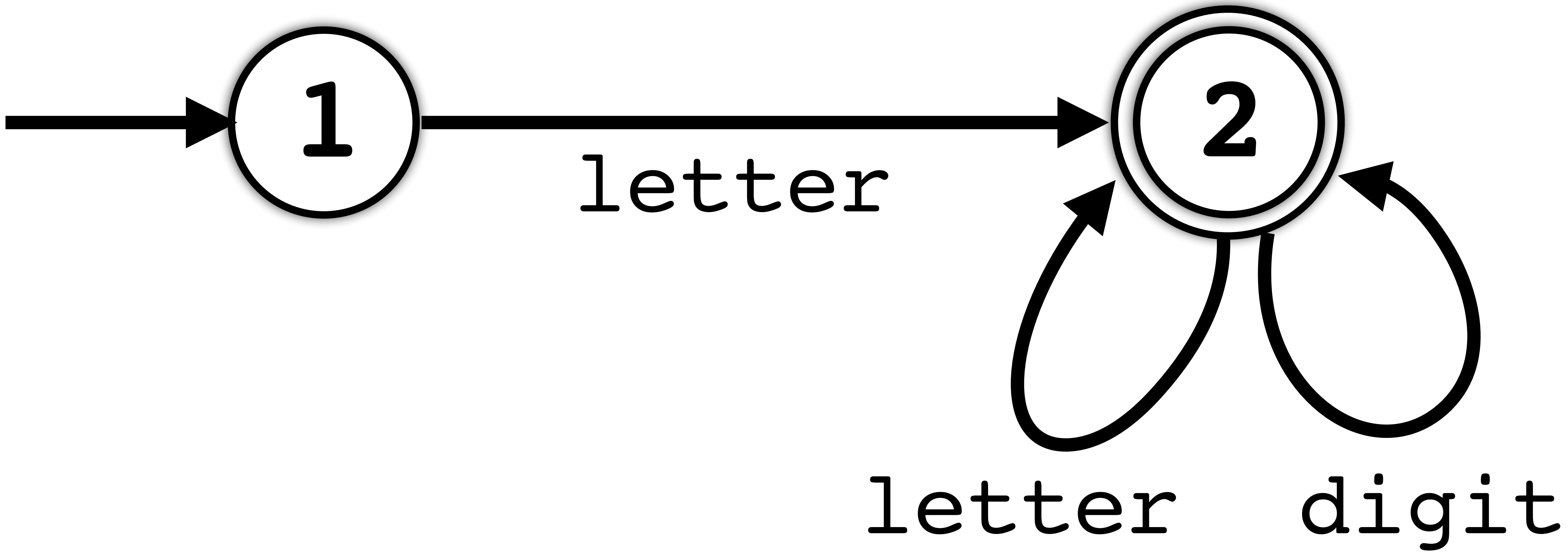
Task 4



- Show the sequence of transitions that this NFA goes through when matching the strings below. If there is more than one possible path, show them all.

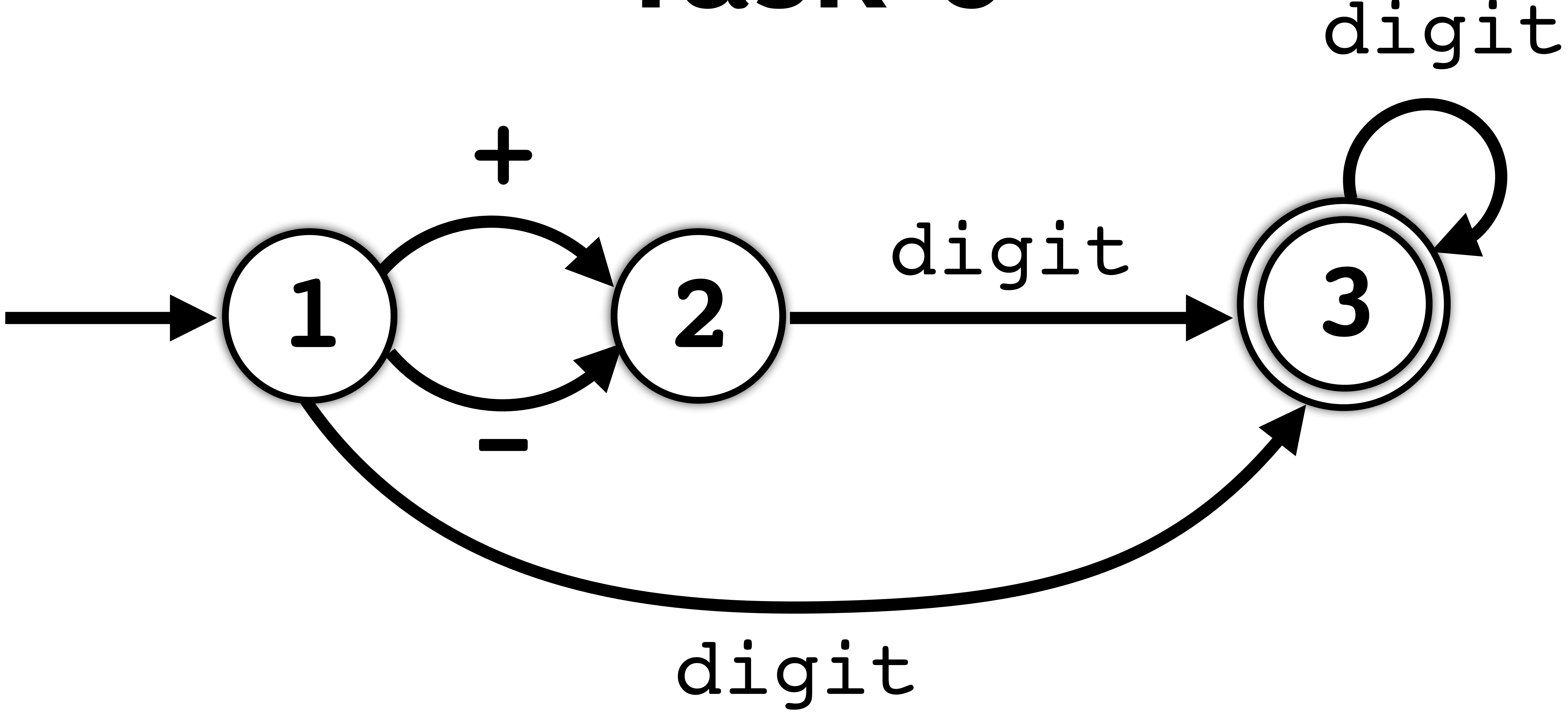
1. "ab"
2. "aab"
3. "abb"
4. "abbb"

Task 5



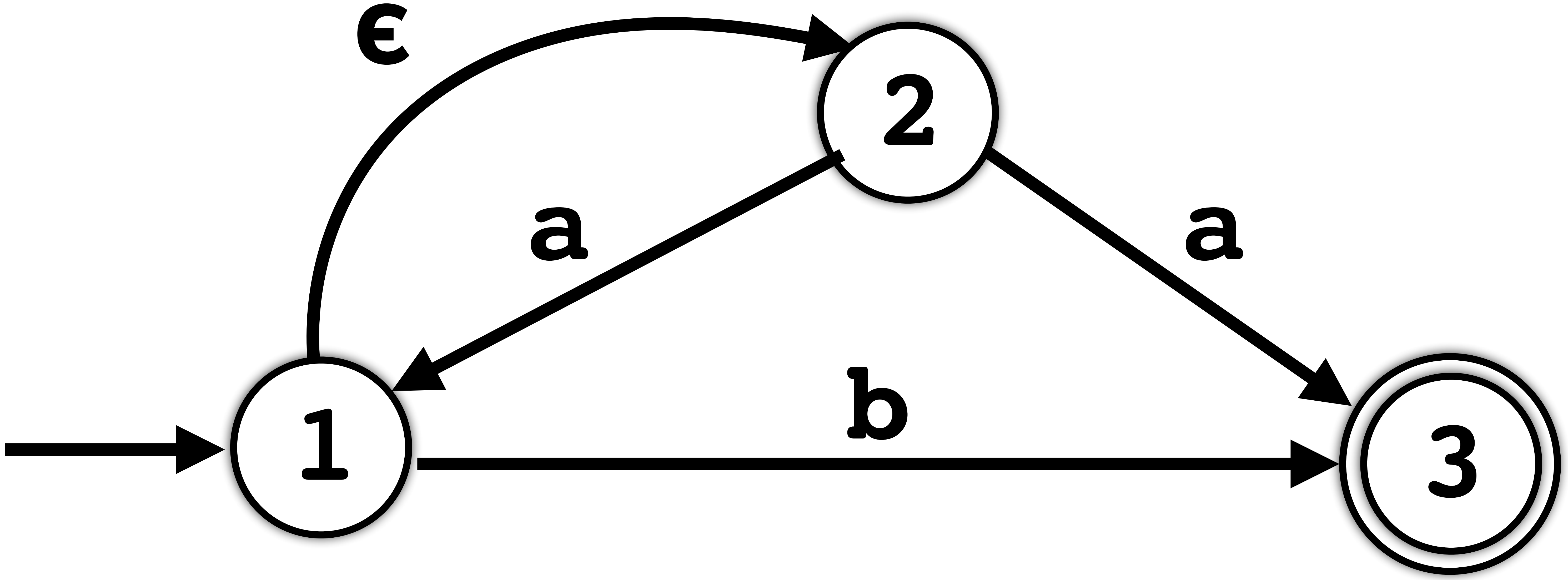
- Implement a lexical analyzer that returns 'true' for a legal identifier, false otherwise.
    1. Modify `IdentS_template.java` using the 'hardcoding" method
    2. Modify `IdentI_template.java` using the 'softcoding" method

Task 6



- Implement a lexical analyzer that returns 'true' for a legal signed integer, false otherwise. Use both hard- and softcoding.

Task 7



- What strings does this NFA accept?

Task 8

- Find a regular expression that is equivalent to this regular grammar.

$$\begin{array}{ll}
\mathtt{S} & \rightarrow \ \mathtt{X}\ \ \mathtt{Y} \\
\mathtt{X} & \rightarrow \ \underline{\mathtt{a}} \ \mid \ \underline{\mathtt{b}} \ \mid \ \underline{\mathtt{c}} \\
\mathtt{Y} & \rightarrow \ \underline{\mathtt{d}} \ \mid \ \mathtt{Y} \\
\mathtt{Y} & \rightarrow \ \underline{\mathtt{e}} \ \mid \ \boldsymbol{\epsilon}
\end{array}$$

Task 9

- Find a regular grammar that is equivalent to this regular expression.

```
[a-d,m]+(e|f)?g+
```

Task 10

1. Make up a complicated **regular expression** R (you have to understand it yourself, though!).

2. Share R with your friend.

3. Ask your friend to explain R to you.

4. As your friend to give some examples of strings that the regular expression accepts.

Task 11

1. Make up a complicated **regular grammar** G (you have to understand it yourself, though!).

2. Share G with your friend.

3. Ask your friend to explain G to you.

4. As your friend to give some examples of strings that the grammar accepts.

Task 12

- You're solving the New York Times crossword puzzle. The clue is *"Britain's historic volunteer cavalry"*. This is what you have:

Y			M	A			Y

- Give a regular expression that searchers through
  - `/usr/dict/words` (on Linux)
  - `/usr/share/dict/words` (on MacOS)

for possible answers, like this:

```
> egrep '(^RE$)' /usr/share/dict/words
```

Task 13

- How many words in
  - `/usr/dict/words` (on Linux)
  - `/usr/share/dict/words` (on MacOS)

have between 4 and 5 consecutive a:s and b:s?

- For example, the word "grabbable" has 5 consecutive a:s and b:s.
- Use the "x{3,4}" regular expression syntax.

```
 > egrep '(^RE$)' /usr/share/dict/words
```

Definitions and Algorithms

```
state := start state
c := first char
while (true) {
   switch (state) {
      case state_1: {
         switch (c) {
            case char_1 : {
               c := nextChar();
               state := new state;
            }
          }
       }
       case state_2: {
           ...
       }
       ...
  }
}
```

Set the start state and get the first input character

Switch over the current state …

… and the current character

Move to the next character and select a new state

```
state := 1
c := first char
while (not ACCEPT[state]) {
    newstate := NEXTSTATE[state,c]
    if ADVANCE[state,c] {
        c := nextChar()
    }
    state := newstate
}
```

Set the start state and get the first input character

Are we done?

Should we move to the next character?

Move to the next state

collberg.cs.arizona.edu
ligerlabs.org

Supported by
NSF SATC/TTP-1525820
SATC/EDU-2029632