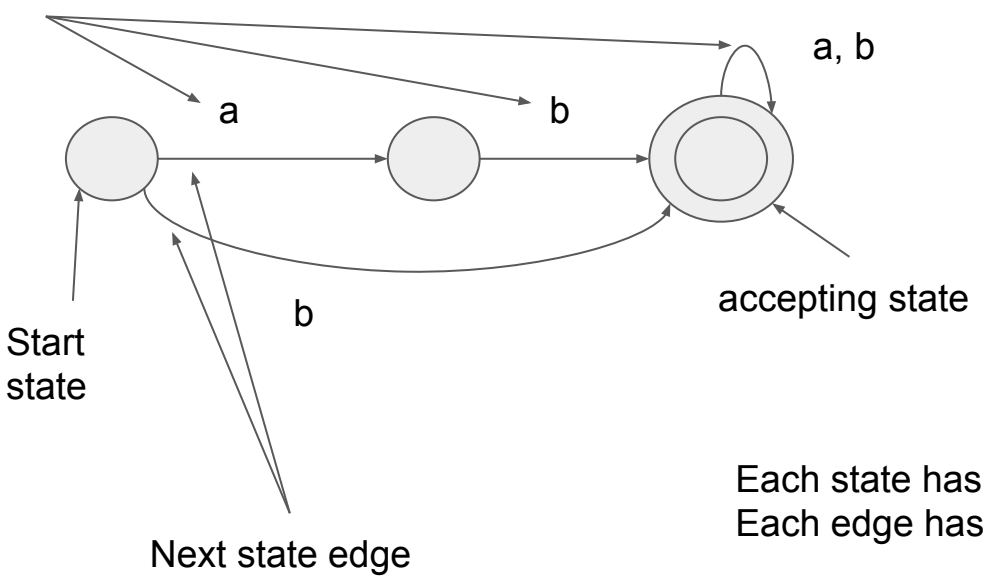
Here is your regular state machine

Edge labels



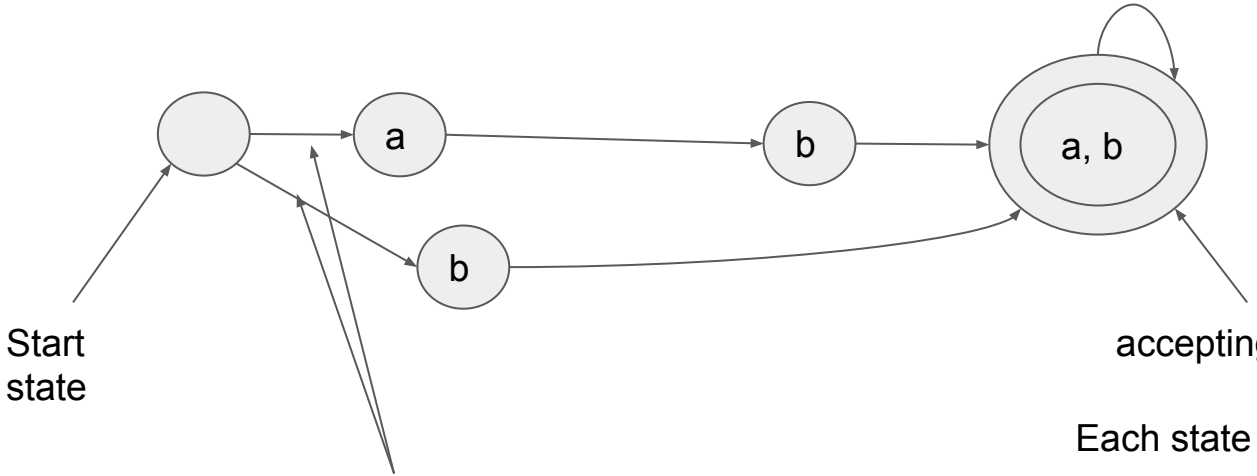
Each state has edges to the next states
Each edge has a label on it

New idea for a state

What if everything was a state?

New idea for a state machine

What if everything was a state?



Each state has edges to the next states

New idea for a state

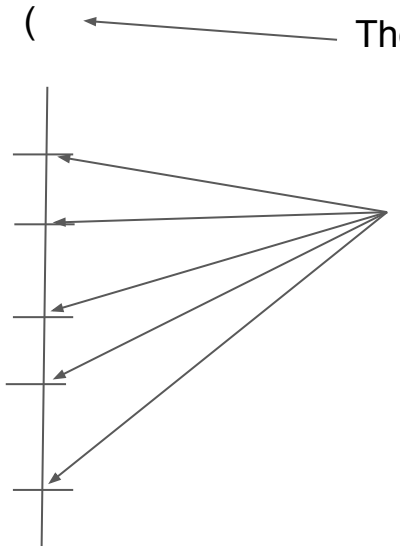
What if everything is a state?

What if each state runs a function, and the function doesn’t necessarily have to do a character check?

What if you could visit the same state at different times and do something different each time?

New idea for a state

The state name is a sequence of strings(each string can have spaces)



The different possibilities of what the second string can be

You can make as many second strings as you want

Examples of first string, second string:

(, one
(, a different string
(, 3
(, the 4th string

Definition of the state

Name
- This is a sequence of strings

Next states
- A sequence of state names that are options coming after the current state and the current state's substates have been run
- Evaluated Like an if else if else sequence that comes after the current state

Children
- A sequence of state names that are options coming after the current state as substates
- Each child state is the start substate of a graph of substates

- This is so you can make a hierarchy of states

Parent
- Metadata

Functions
- 1 function for each second string in the name
- The function doesn't necessarily have to do a character match

Just as a note:
- The state machine is a graph defining the flow of control. You will run a function when you visit the state

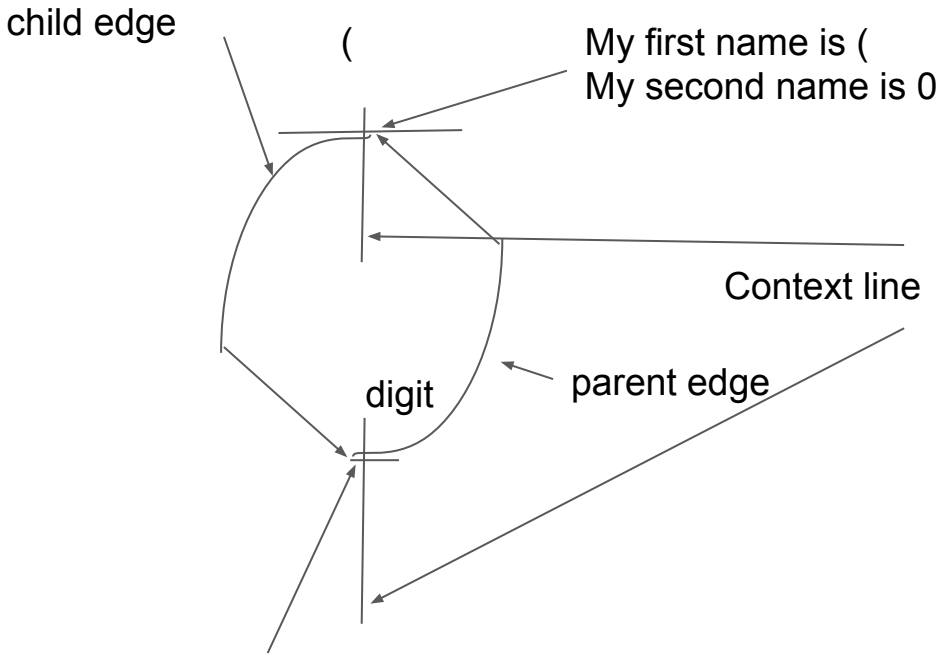
New idea for a state

What if everything is a state?

What if you could visit the same state at different times and do something different each time?

What if you wanted a hierarchy of states?

New idea for a state(the functions are not shown)

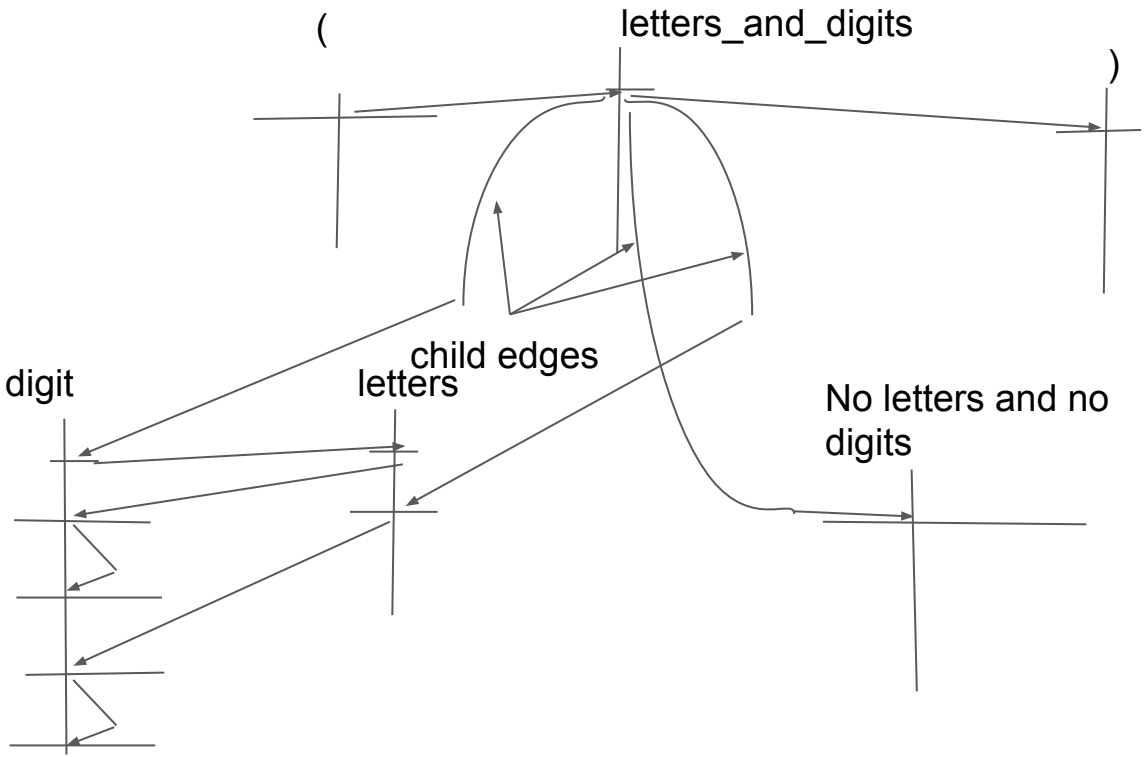


Each (state, context) can run a function

The function always returns true or false

The function can run any code before returning true or false

New idea for a state hierarchy



We want to recognize the below sequences
(digit letters digit digit)
(letters digit digit)
( )

Lets see an example

First, evaluation style

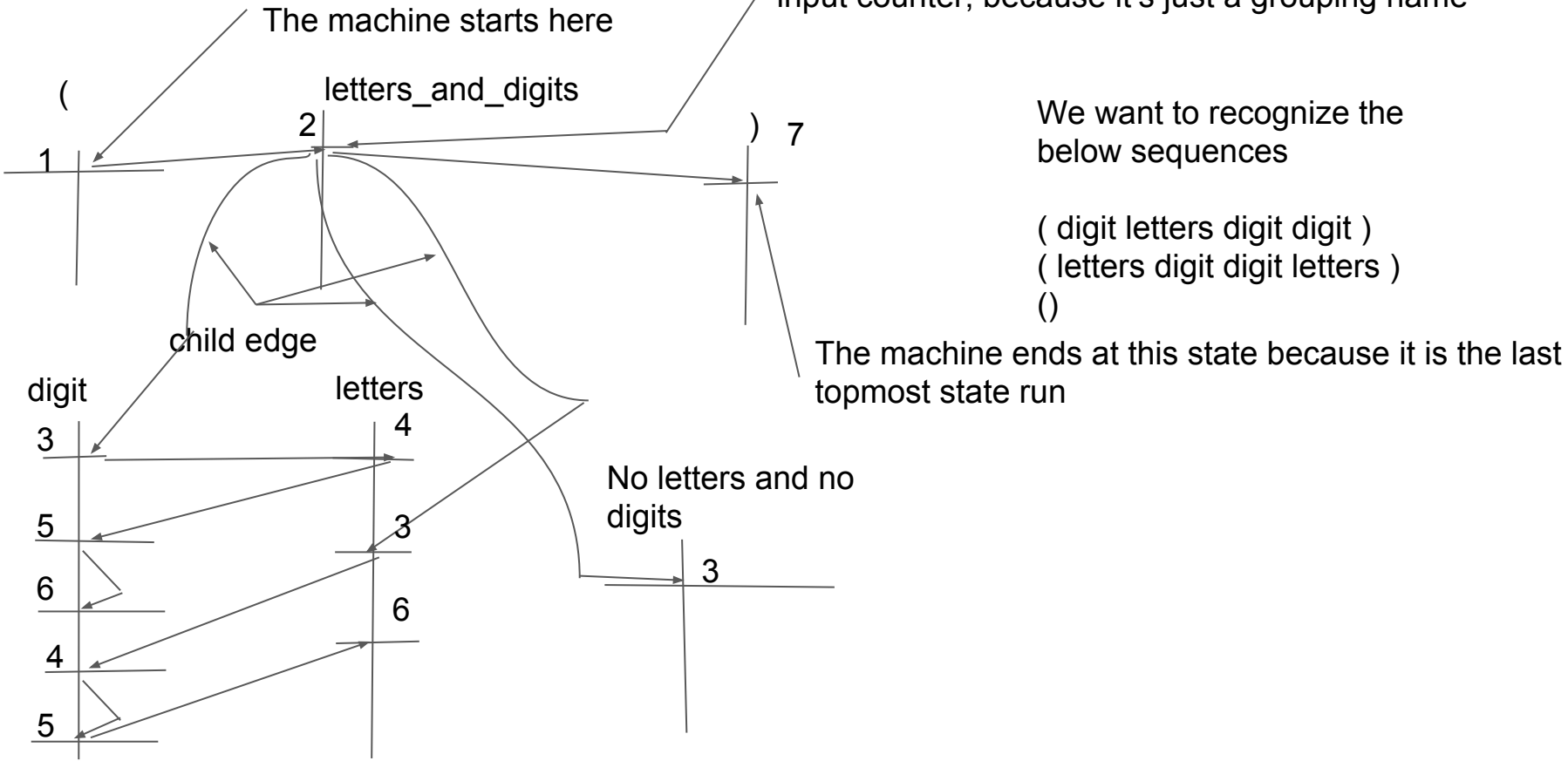
Each next state is run as options like a sequence of if else if statements

If

Else if

else

Here is the order the states are run in



We want to recognize the below sequences

(digit letters digit digit)
(letters digit digit letters)
()

The machine ends at this state because it is the last topmost state run

Here is the layout of the code structure for a state

```
Name (first string)

      Next
            (second string)
            0
                  [‘another_first_string’, ‘another_second_string’], [‘another_first_string_2’,
            ‘another_second_string_2’]

            names\’s second context
                  [<a [state, case] would go here>]

      Children
            (second string)
            0
                   [‘a child’, ‘the child\’s case’]

      Functions
            (second string)
            0
                  functionName
            names\’s second context’
                  anotherFunctionName
```

Let’s see the actual code for 1 state

Quick notation for the below example: (state, case)

This is what the syntax looks like for 1 state with 2 cases and 1 child

```
['name', [
      ['next',
            [['0', [[‘another_name’, ‘another_case’], [‘another_name_2’, ‘another_case_2’]]]
            [‘names\’s second context’, [ <a [state, case] would go here> ]]       ]]],
      ['children',
            [['0', [ [‘a child’, ‘the child\’s case’] ] ]]],
      ['functions',
            [['0', functionName ],   [‘names\’s second context’, anotherFunctionName]   ]]]],
```

Common mistakes for typing this
Having ‘[’ or ‘]’ in the wrong place

The actual functions are defined elsewhere

Let’s see the code for a function the state runs

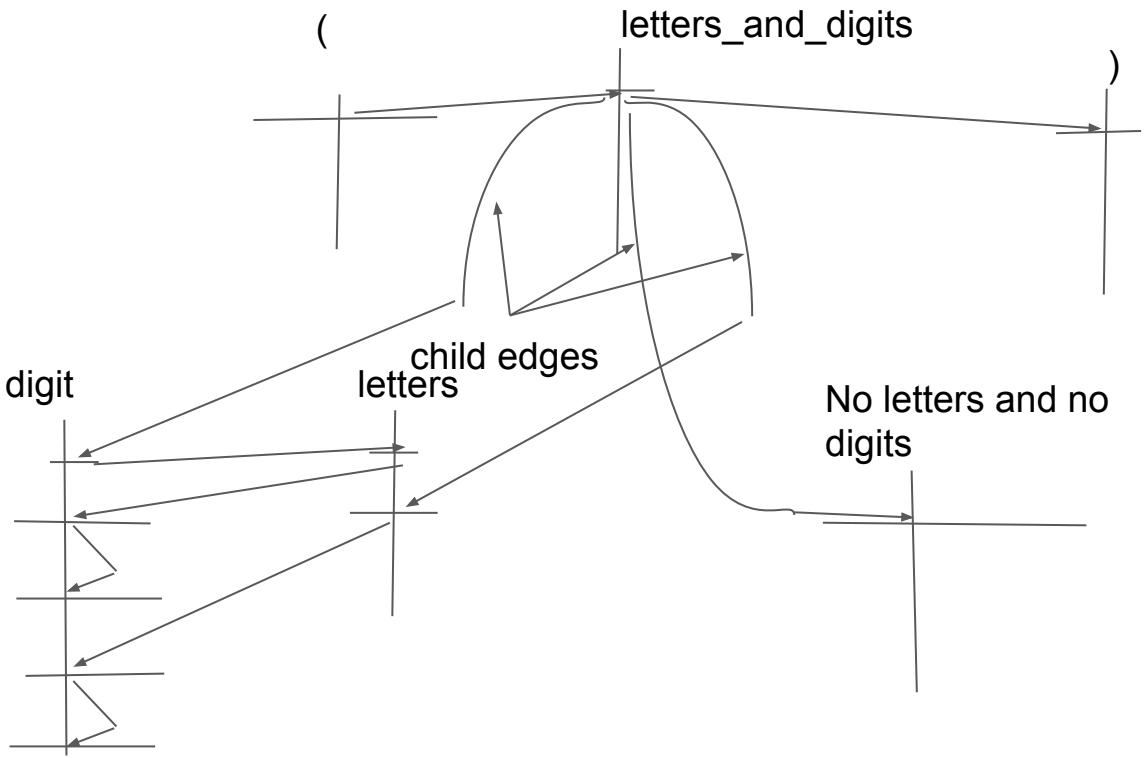
```
def validOp(node, var_store):   # a main (state_string_1, state_string_2) function signature, not a helper
function
      i = var_store['i']
      input_ = var_store['input']
      if isOp(node, var_store):
            var_store['operation_vars']['a'] = input_[i - 1]
            return True
      return False
```

Make sure all the connections are set the way you expect before you test it

When you design the sequences write the the states as terms in a series next to the “next” and “children” names within the state attributes in the code for each state
(string_1, string_2) -> (another_string_1, another_string_2)

That way you know what order you want

Diagram for simple state machine



Link to github page for project materials

https://github.com/dtauraso/State-Machine-Tutorial