CS112 (LFA) - Projects summary

April 2021

1 Lab 1

Exercise 1. *(1p) Implement a library/program in a programming language of your choosing to load and validate a DFA input file.*

```
dfa_parser_engine.py dfa_config_file
```

Exercise 2. *(2p) Implement a library/program in a programming language of your choosing to test acceptance of a DFA - loaded from a DFA config file.*

```
dfa_acceptance_engine.py dfa_config_file <word_to_test>
```

2 Lab 2

Exercise 3. *(2p) Implement a library/program in a programming language of your choosing to test acceptance of an NFA - loaded from an NFA config file (similar to the DFA config file from exercise 1).*

```
nfa_acceptance_engine.py nfa_config_file <word_to_test>
```

Exercise 4. *(2p) Implement a library/program in a programming language of your choosing to convert an NFA - loaded from an NFA config file - to a DFA.*

```
nfa_dfa_conversion_engine.py nfa_config_file
```

3 Lab 3

Exercise 5. *(2p) Implement a library/program in a programming language of your choosing to convert a DFA to a **minimized** DFA.*

```
dfa_minimization_engine.py dfa_config_file
```

4 Lab 4

Exercise 6. *(2p) Implement a library/program in a programming language of your choosing to simplify a CFG (see next page for CFG input file example). The program should:*

- *remove useless productions (CFG reduction)*
- *remove unit productions*
- *remove null productions*

```
cfg_simplifier_engine.py cfg_config_file
```

CFG input file format:

```
#
# comment lines (skip them)
#
Start:
    S
End
#
# comment lines (skip them)
#
Epsilon:
    0
End
#
# comment lines (skip them)
#
Terminals:
    a
    b
    c
    ...
End
#
# comment lines (skip them)
#
Nonterminals:
    A
    B
    C
    ...
End
#
# comment lines (skip them)
#
Productions:
    S  ABAC
    A  aAb|0
    B  bC
    ...
End
```

5 Lab 6

Exercise 7. *(1p) Create and document a structure for a* ***TM configuration file****. Implement a library/program in a programming language of your choosing to load and validate a* ***TM configuration file*** *based on your own structure.*

```
tm_validation_engine.py tm_config_file
```

Exercise 8. *(2p) Implement a library/program in a programming language of your choosing to load and validate a TM and run it against an input. Use the TM and input as shown in* ***Sipser - 3.8*** *for testing.*

```
tm_simulator.py tm_config_file input
```