```
Task documentation DKA: Determination of finite automata in Python 3
    for IPP 2013/2014
Name and surname: Mark Birger
Login: xbirge00
```

Objectives

Common task was to create module in Python 3 language, which can process finite state machines *(FSM below)*. Processing includes parsing, removing empty states, determinization and valid printing *(also string processing as an extension)*. Additional task was special parameters such as case-insensitive processing.

Development

At the start, i estimate the size go the project. Because Python is heigh-level programming language and provides a lot of useful features for current task, sets operations for examples, estimated script size was 500 lines. I selected iterative development strategy, with testing on each complete part of the task. Coded on local machine, because Python is absolutely cross-platform language. Only at the finish, code was tested at the merlin server. Applied test was extended by my own FSM samples.

Script uses object oriented paradigm. Main class named Automata provides set of methods applicable for real FSM. *Full description in design section.*

At task understanding and planning i've spent **1 hour**. Development of the first version, covering minimum task took **6 hours**. I've spent **18 hours** for searching of errors, refactoring non-pythonic constructions and bug-fixing.

Design

1. Parsing input arguments, opening file/stdin.
2. Main function creates entity of `class Automata`. With constructor sends description of FSM.
3. Constructor setup basic class attributes, calls `parseDesc()` method.
4. `parseDesc()` method erase comments, whitespaces, parse parts of FSM using regular expressions. Then calls `parseStates()`, `parseAlphabet()`, `parseRules()`, `parseInitial()`, `parseFinal()`, methods, which parse different parts of FSM.
5. `checkSemantic()` method checks semantic of parsed FSM. If everything is ok, parsing is finished at this stage.
6. Main calls `removeEmpty()` method for created Automata entity. It uses `emptyClosure()` method and removes empty rules with IFJ presentation algorithm.
7. Main calls `determinize()` method. It determinize states using IFJ algorithm.
8. Rewritten `__str__()` method allows to print Automata entity using standard `print()` function.

Also i used `prepareSymbol()` and `newState()` methods with `@staticmethod` decorator, to use it without self dependency.

Task ambiguity

Task not contains problematic moments. But rewriting pseudocode needs to pay a lot of attention. For example, in removing of empty rules part it needs to check alphabet symbols till finite state creation.

Another choice was using ragular expressions, because comments block can be at every position and parsing using another FSM is painfull conslusion in this case.

Complex aspects

When i started this project, i already known Python for two years. It wasn't problem to combine different types like sets, tuples, list and dictionaries. It was right language choice for this project.

Rules set datatype. First choice was dictionary with `set(state,symbol)` and `new-state` accordance. But after i changed it to a list of tuples structure. It was needed because initial FSM wasn't determined at some states and dictionary doesn't support same keys. Set properties provided by permanent conversion to set datatype and back.

Set of sets. Another hard-task was rewriting of recommended algorithms. This algorithms uses set of sets operations. In Python set datatype is mutable and not hash-able. Because of this i used `frozenset` datatype at lower level.

Three keys sorting. This problem solved by using `lambda` function as a key, which provide tuple of three values for sorting function. Lambda functions is very useful Pythonic way to solve small tasks.

Class attribute names. It was choice of different sets names between same as in provided IFJ presentations and **PEP8** recommendation (lower case in class attributes). I selected IFJ-like names to save similarity with IFJ pseudocode.

Conclusion

This project introduced to me difficult aspects of Python programming language such as lambda functions, sets and OOP. I've understand object oriented paradigm because of good similarity with FSM models. Also i've understood better algorithms of FSM determination, which is basics of formal languages theory.

Metrics: 473 lines; 16542 symbols.