Rheinisch-Westfälische Technische Hochschule Aachen
Chair of Process and Data Science
Prof. Dr. Wil van der Aalst

Process Conformance Checking in Python in SS 2021

Alignments on NFA(s) in Micropython

Group:3

Sprint 3

9.07.2021

supervisor: Alessandro Berti

Contents

List of Figures

1 Introduction

The goal of this project is to implement conformance checking in python with no dependencies. In order to achieve it, in the first sprint we started with converting regular expressions into NFAs, implemented NFA model and checked its fitness. In the second sprint alignments were computed, unit tests were developed and code of first sprint was optimized by catering the edge cases. This sprint is about improving the implementation by adapting to a new and more optimal approach in terms of memory and performance. Chapter 2 discusses various aspects like work distribution, improved optimal alignment computation, restructuring of the code base, documentation and testing. Chapter 3 is comprised of phase reviews of the entire team.

2 Summary of Sprint 3

2.1 Work Distribution

The tasks of this sprint were divided into the following major phases:

- Theory Analysis
- Implementation Improvements
- Restructuring of Code Base
- Documentation
- Adding project manual (README) in the repository
- Testing

The Trello board shows the development cycle for this sprint.

2.2 Implementation Improvements

In the last sprint we implemented a working solution to compute alignments on nfas for a given trace. The goal for this sprint was to improve the implementation. We did this by implementing a new approach to search through the big search space with the dijkstra algorithm more efficiently. As defined in the requirement engineering part of our project, we used the two measurements memory and performance to evaluate the improvements.

2.2.1 Improved Optimal Alignment Computation

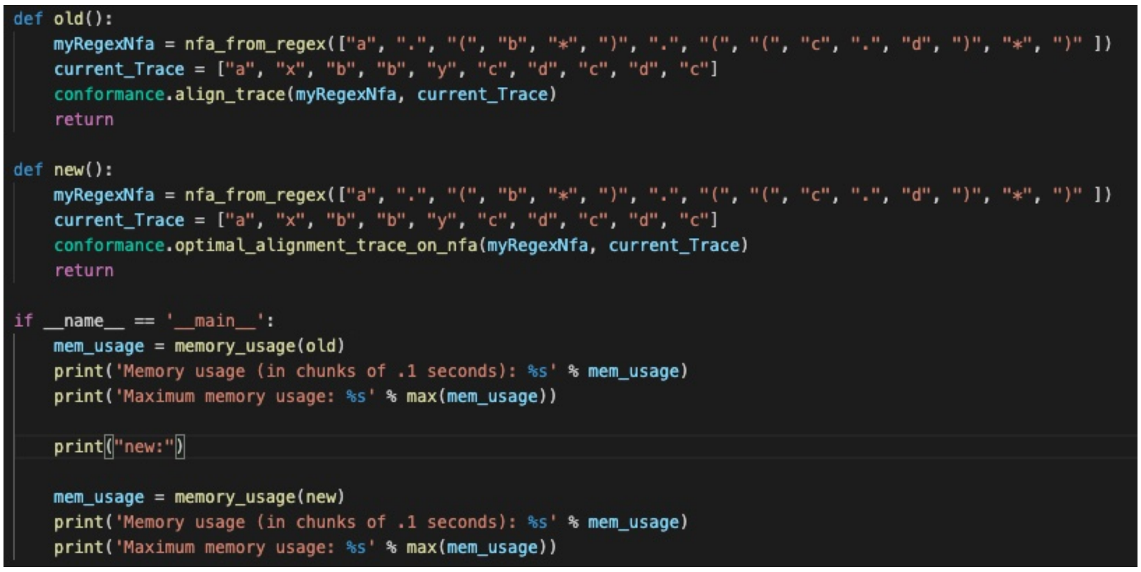
The improved optimal alignment algorithm is still conceptually the same as described in sprint 2. We are still using the dijkstra algorithm and the same cost function as before. The improvement is that we do not compute the combined nfa of the trace nfa and the model nfa anymore. By not computing this combined nfa we are able to save memory and increase the performance.

The purpose of the combined nfa was to be able to search the search space easily by using the nfa as the search space itself. Now we instead search in the search space by computing possible neighbours of an already found node only when necessary and we do not store this information in memory.

Moreover we now use more memory efficient ways to store the book keeping information that are necessary for the dijkstra algorithm. For example we use immutable objects when possible as the can be processed more efficient.

Evaluation of the improvements:

We evaluated the **memory usage** and performance of the old and new approach. The library memory-profiler was used to get insides in the memory consumption of the two approaches. The code used to measure the memory usage is the following:

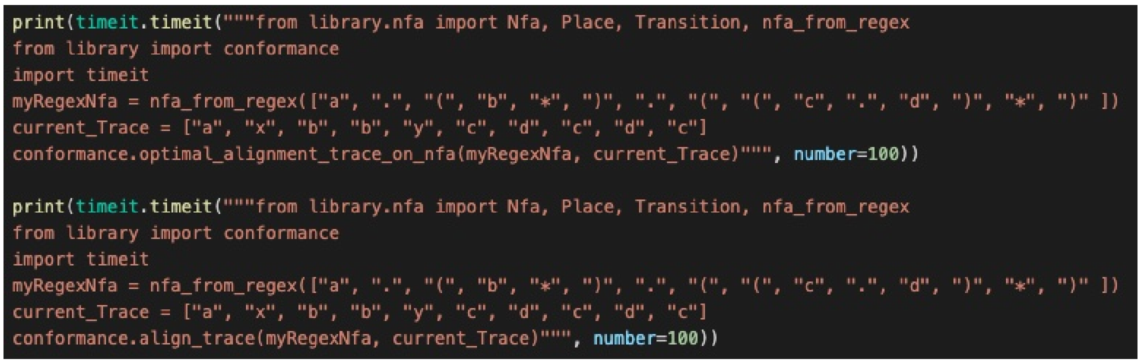


```
def old():
    myRegexNfa = nfa_from_regex(["a", ".", "(", "b", "*", ")", ".", "(", "(", "c", ".", "d", ")", "*", ")" ])
    current_Trace = ["a", "x", "b", "b", "y", "c", "d", "c", "d", "c"]
    conformance.align_trace(myRegexNfa, current_Trace)
    return

def new():
    myRegexNfa = nfa_from_regex(["a", ".", "(", "b", "*", ")", ".", "(", "(", "c", ".", "d", ")", "*", ")" ])
    current_Trace = ["a", "x", "b", "b", "y", "c", "d", "c", "d", "c"]
    conformance.optimal_alignment_trace_on_nfa(myRegexNfa, current_Trace)
    return

if __name__ == '__main__':
    mem_usage = memory_usage(old)
    print('Memory usage (in chunks of .1 seconds): %s' % mem_usage)
    print('Maximum memory usage: %s' % max(mem_usage))

    print("new:")

    mem_usage = memory_usage(new)
    print('Memory usage (in chunks of .1 seconds): %s' % mem_usage)
    print('Maximum memory usage: %s' % max(mem_usage))
```

Figure 1: Evaluation of memory usage improvements.

We saw that the old approach given the regular expression created $0.2MiB$ in the memory where the new approach only created $0.0004MiB$. This shows that the improvements indeed lead to way smaller memory requirement for computing alignments.

In addition, we evaluated the **performance** increase of the new implementation by measuring the execution time of one hundred alignment calculation. The python package timeit was used in order to evaluate the execution time. To have a fair comparison both alignment methods were given the same regex to compute an nfa and an alignment for a trace. The code for that analyses can be found in the following figure:



```
print(timeit.timeit("""from library.nfa import Nfa, Place, Transition, nfa_from_regex
from library import conformance
import timeit
myRegexNfa = nfa_from_regex(["a", ".", "(", "b", "*", ")", ".", "(", "(", "c", ".", "d", ")", "*", ")" ])
current_Trace = ["a", "x", "b", "b", "y", "c", "d", "c", "d", "c"]
conformance.optimal_alignment_trace_on_nfa(myRegexNfa, current_Trace)""", number=100))

print(timeit.timeit("""from library.nfa import Nfa, Place, Transition, nfa_from_regex
from library import conformance
import timeit
myRegexNfa = nfa_from_regex(["a", ".", "(", "b", "*", ")", ".", "(", "(", "c", ".", "d", ")", "*", ")" ])
current_Trace = ["a", "x", "b", "b", "y", "c", "d", "c", "d", "c"]
conformance.align_trace(myRegexNfa, current_Trace)""", number=100))
```

Figure 2: Evaluation of performance improvements with the timeit package.

The result is that the new execution needed 0.289813806 seconds and the old version needed 7.917175938 seconds. Therefore the new implementation runs **27 times faster** then the old one. This can bee seen as a big improvement and we are happy with the result.

2.3 Restructuring of the code base

To make the project easier to maintain we restructured the code base. The structure of our code base represents now clearly the functionalities of our library. Therefore we now have to separate files. One of them only contains the nfa model related code and the other one is concerned with the conformance calculations. The names were choosen accordingly *nfa.py* and *conformance.py*.

In addition, we added an example file that demonstrates the usage of the library we created. Here we show how to import the functionalities and how to use them. We give examples for:

- Creating an nfa
- Convert a regular expression into an nfa
- Check whether a trace perfectly fits an nfa
- Calculate the percentage of traces in a log that perfectly fit the nfa
- Compute an alignment for a trace and an nfa
- Compute alignments for each trace of a given log

2.4 Documentation

For documenting the code, we have used Pdoc. It is a software package for generating API documentation for Python language. Documentation is generated from the source code docstrings and HTML documentation is generated for chosen Python modules. The html files are uploaded in the Git repository and can be accessed by going to *html - nfa.html, conformance.html.*

Project manual (README) is also added to guide the users regarding the details of our project like its purpose, why is it useful, what to refer to in case of any ambiguities. Its a brief overview of the overall idea and the details of its execution to make it easy for the end users to not only get the idea of our work but also enable them to use it in future, wherever required.

2.5 Testing

Unit testing is a software testing method by which individual units of source code sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine whether they are fit for use. Unit tests are typically automated tests written and run by software developers to ensure that a section of an application (known as the "unit") meets its design and behaves as intended. In this sprint we used python "unittest" framework for unit testing. The "unittest" unit testing framework has a similar flavor as major unit testing frameworks in other languages. It supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework. As our implementation has been separated in two files one file nfa.py that contains the implementation of the NFA data structure class and all its helping functions and the other file conformence.py that contains the implementation of the optimal alignment algorithm so our unit tests were also separated into two testing file as follows: test_nfa.py: This file contains the TestNfa class which is the main unit testing class that covers all test cases of the main modules and units in the NFA main data structure class and all it helping functions. test_conformence.py : this file contains TestConformence class that contains all unit test for the mail optimal alignment algorithm implementation. The unit testing this sprint covers all test cases for almost every single function in the two main classes, and I believe we now have a reliable and robust implementation.

3 Phase Review

3.1 Asad Tariq

The goal of this sprint was optimization of our technique for alignment computation and was the most important sprint. The team approached this sprint with the right level of communication and task distribution. My main task for this sprint included research for optimal ways of implementation of our technique and setting up of input check functions. The team remained very cooperative and supportive in the entire sprint. Every one played their part and got done with their tasks on time.We had a good communication and we tried to handle the issues we faced collectively.Overall, I got the opportunity to learn a lot from this lab and also from my fellow members. It strengthened my technical abilities and also taught me task prioritization. I learned time management and stress management which will prove to be beneficial for my future.

3.2 Mahmoud Emara

This sprint was as always, a success, one of the key factors for the successful completion of this sprint was the daily update that we made, this helped each one of us to stay connected and informed about every change. My main task allocation for this sprint was to complete the implementation of the unit tests since we have added new functions and modified the old version of our code to be more optimized in terms of performance and memory, moreover, I had to adjust and separate the unit tests into two separate files one for each main implementation file to keep our project more readable and testable. For the assessment part, I am pretty sure that everything will run smoothly since we made sure that everything works in a perfect manner. Everything went well thanks to the tremendous help from each group member.

3.3 Syed Faizan Hassan

In this sprint, the main task was optimization of our already implemented technique. We had a meeting on the first day of the final sprint where the tasks were divided.Three of us worked on the theoretical aspect and looked for how our approach can be optimized. Whereas, the remaining two group members were assigned to documentation and testing task.Then we conducted a meeting with the professor to discuss our ideas in order to improve our implemented approach.We extended our approach with another input method i.e directly providing a log (list of traces) and also a log input check function for it. In the entire sprint, we communicated well and tried to resolve the problems we faced. I am very happy with the way our team worked during the whole project and such good coordination is the reason we were able to complete our project in time.This lab provided me an opportunity to strengthen my already built concepts and improved my technical abilities. This lab served as a great platform for me to not only deepen my knowledge, but also equip myself with more skills, both interpersonal and technical, and experience.

3.4 Lukas Liß

I really liked this sprint because we were able to concentrate on optimizing the implementation. The meeting with Alessandro Berti was very help full to discuss and clarify the approach on how to optimize our implementation. I enjoyed the discussions we had in the team about the algorithm and I think our communication was very good in that regard. This allowed us to experiment with different implementation which I think increased the quality of the end result a lot. From my point of view our teem work got better every sprint so that I can only say positive thinks about the team dynamic we had in this sprint. I liked especially that everyone had a clear role but at the same time we all supported each other when a problem occurred. I learned a lot during this sprint but also in the course of this project in general. Not only did my python skills improve but especially the soft skills required to finish such a projects improved. I am very much looking forward to finalize the project with this team because it was a pleasure to work with all of them.

3.5 Mina Khalid

This sprint was dedicated to improve the alignment computation technique in terms of memory and performance. We had a discussion with Mr. Berti on the same which turned out to be really fruitful in terms of getting ideas that we had overlooked in our previous sprint. It gave us a new perspective for improving our approach over the defined parameters, as committed in the beginning of this project. We then had multiple discussion sessions within our group and all the team members contributed for optimizing the approach further. It was a great pleasure working with this team. Everyone was readily available wherever anyone got stuck. This project provided me with an amazing learning experience not only in terms of technicalities and domain knowledge in the field of conformance checking but also in personal management skills.

References

[1] Wil van der Aalst. Data Science in Action. Springer Berlin Heidelberg, 2016.

[2] Rabin, M. O. and Scott, D. Finite Automata and Their Decision Problems.IBM Journal of Research and Development, 1959, doi: 10.1147/rd.32.0114

[3] Alfred Vaino Aho; Monica S. Lam; Ravi Sethi; Jeffrey D. Ullman (2007). "3.7.4 Construction of an NFA from a Regular Expression" (print). Compilers : Principles, Techniques, Tools isbn: 9780321486813

[4] Burge, W.H. (1975). Recursive Programming Techniques. isbn: 0-201-14450-6

[5] R. Rastogi, P. Mondal and K. Agarwal, "An exhaustive review for infix to postfix conversion with applications and benefits," 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015, pp. 95-100.

[6] https://en.wikipedia.org/wiki/Thompson