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 2

18.06.2021

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. However, this second sprint is dedicated for computation of optimal alignments, development of unit tests and optimization of the code of first sprint by catering the edge cases. This document covers a brief summary of what we had planned and how we have achieved it in the current sprint. Chapter 2 discusses various aspects like work distribution, implementation of the planned tasks, optimal alignment computation, input checks, documentation and testing. Chapter 3 is comprised of phase reviews of the entire team.

2 Summary of Sprint 2

2.1 Work Distribution

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

- Theory Analysis
- Computation of Optimal Alignments
- Development of Unit tests
- Optimization of sprint 1 code in terms of handling edge cases
- Bug Fixing
- Documentation

The Trello board shows the development cycle for this sprint.

2.2 Implementation

As proposed already in the sprint review of the last sprint, we focused on computing optimal alignments in this sprint. Additionally we implemented unit tests and improved the implementation especially for edge cases for the already existing code from sprint 1.

2.2.1 Optimal Alignment Computation

Given a de jure model and a trace one can try to align them. An alignment is a sequence of moves. We differentiate between three types of moves:

- **Synchronized Moves:** Were the activity of the model is equal to the one that happens in the trace.
- **Move on model only:** Here the de jure model performs an activity that is not present in the trace.
- **Move on log only:** Here there happens an activity in the trace of a log wheras nothing happens in the model.

To compute an optimal alignment there need to be a cost function to determine in which alignments are better than others. As advised by our stakeholders we used the following cost function:

Move in alignment	Cost of the move
Synchronized Moves	0
Move on model only	1
Move on log only	1

Table 1: Cost Function for alignments.

As the most common approach for calculating alignments are based on petrinets, we had to adopt t the algorithm to fit our use-case of calculating optimal alignments based on nfas.

Our approach to calculate an optimal alginment for a given trace and a de jure model as an nfa is as follows:

1. Check whether the trace is fitting on the given model. The reason we do this is to save resources. The optimal alignment of perfectly fitting trace is trivial to compute as it only consist of synchronized moves. When the trace is perfectly fitting we simply return this easy optimal alignment. In the case the trace is not perfectly fitting based on the model we continue with step 2.

2. Create a nfa for the trace. In this step we create an nfa that only accepts the trace as its language and nothing else. This can be done by having a start and end place and n-1 places in between where n is the length of the trace. Then we can add n transitions between these places from start to end place. These transitions require exactly one element of the trace in the correct order.

3. Combine model nfa and the created trace nfa. An correct alignment needs to bring the de jure model nfa to an accepting state as well as the created trace nfa. Therefore we want to create a new nfa that simulates the parallel execution of both nfas. Lets say the de jure model nfa has the places $p_{model_1}...p_{model_n}$ and the trace nfa has the places $p_{trace_1}...p_{trace_m}$.

Then the combined nfa has a place for each element of the cross prdouct of these places. So the combined places are $p_{trace_i,model_j}$ with $i \leq m$ and $j \leq n$.

The start state is the state that references both start states of the original nfas. The endplaces are all places where both referenced places in the two original nfas are accepting places.

Then we can add all the synchronous moves by looking in the two original nfas for all the possible synchronous moves. We add a transition into the combined nfa that has the start place that matches the two start places the transitions had in the two original nfas. The cost of these transitions is 0 regarding to Table 1.

We can add the move on log or move on model transitions by simply adding them to the combined nfa to all those places in the combined nfa that reference to the matching places. These transitions have a cost of 1 regarding Table 1.

4. Dijkstra on the combined nfa. Dijkstra is a well known algorithm to find a shortest path in a network. We use this algorithm to find the shortest path in the combined nfa. As synchronous moves have no cost, but the others have, it will find the cheapest alignment that ends in a state that statisfies both original nfas. And when both original nfas are accepting the path then the path is an alignment. Therefore we find the

optimal alignment with this method.

Obviously there are still things we can improve in the upcoming sprints. Especially we are planning to improve the handling of epsilon transitions. This should be no problem as epsilon nfas can also be transformed into nfas without an epsilon transition. Moreover, the implementation is not yet optimized for performance but for understand ability. As the goal is to be as good performing and memory efficient as possible, we will focus on this in the upcoming sprint.

2.3 Input checks

Input checks are an essential part of a project in order to alarm the user in case of wrong input. There is a certain convention in which input should be passed to compute the NFA. In our case, the input should be a list of alpha-numeric values for a trace and alpha-numeric values along with some special characters which are predefined for a regular expression. We added the input checks in a way so that the user gets notified with an alarm message containing the respective error.

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 file for sprint 1 is uploaded in the Git repository and can be accessed by going to *library - html - nfa.html*.

2.5 Testing

In computer programming, 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.
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 class. The unit testing in this sprint covers all basic test cases for the main modules in the NFA class we plan to extend this class to handle more advanced and edge test cases in the next sprint to have a reliable and robust implementation.

3 Phase Review

3.1 Asad Tariq

The goal of this sprint was optimal alignment computation and was most important sprint. The team approached this sprint with the right level of communication and task distri-

bution. My main task for this sprint included RD for optimal implementation required to perform the tasks and covering the theoretical concepts. The team remained very co-operative in the entire sprint, we had a good communication and we tried to handle the issues we faced collectively. In the next sprint, I hope to continue with the same level of determination and expect the team to show the same enthusiasm and effort.

3.2 Mahmoud Emara

In this sprint we have reached most of our goals that we set, that is why I think that the second sprint was a complete success. Every group member has done a great job. My main task allocation for the sprint was to implement the unit testing for our main NFA class and make sure that the test cases can be run smoothly on top of that we had to revise some test cases that should be handled in the next sprint. Even though I have Implemented what I was supposed to implement in the last sprint but due to a medical issue, I was not able to push my implementation. One of the key factors for the successful completion of this sprint was the tremendous help that has been offered from the group's members and the willingness to help.

3.3 Syed Faizan Hassan

This sprint, the main task was computation of optimal alignment. We had a meeting on the first day of sprint 2 where the tasks were divided.Three of us worked on the theoretical aspect and looked for the optimal algorithms for computing the alignment. Whereas, the remaining two group members were assigned documentation and testing task. In the entire sprint, we communicated well and tried to resolve the problems faced. Every member delivered their tasks timely. Overall, I am satisfied with the way team performed in this sprint. I look forward to have the same level of cooperation and working spirit in the next sprint as well, with focused implementation of our tasks.

3.4 Lukas Liß

This sprint was way more complex in terms of algorithm creation than the ones before. My main task was to conceptualize and implement the algorithm to compute optimal alignments. I personally really enjoyed the challenge. By discussing the approaches we had in the team we were able to find a good solution and it helped everyone to understand the topic well. I am proud, that the approach we had, works. Moreover, I think we did a great job this sprint and learned from the problems we had at the beginning of the last sprint. The teamwork is still good and everyone is reliable and supportive. I am looking forward to finish this interesting project with this team. For the next sprint I think we should try to keep up the good level of communication and work together to extend the testing and the quality of the implementation.

3.5 Mina Khalid

This sprint was one of the most important sprints so far as we worked on the core task of computation of optimal alignments. We considered all the possibilities for coming up with the most efficient implementation and managed to get this done in due time. All the group members were highly motivated throughout and were up for supporting other

members wherever it was required. I really like the positive attitude of our team and the willingness to go an extra mile whenever needed.

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