Master Thesis

A Haskell Web-Application for Data-Mining Competition-Results from StarExec

A thesis submitted to attain the degree of
Master of Science at the Leipzig University of Applied Sciences
(M.Sc. HTWK Leipzig)

presented by
Stefan von der Krone

B.Eng. in Media Technology, University of applied Sciences Mittweida

born on 19th of April, 1984

citizen of Germany

accepted on the recommendation of Prof. Dr. Johannes Waldmann

2014

Summary

…

Table of Contents

1 Introduction

This is a guide to the Star-Exec-Presenter, a web-based visualization and data-mining software developed in preparation of this thesis. The Star-Exec-Presenter is intended to be an efficient to use software for members of the termination community of the StarExec service, as well as other communities or users of StarExec. The web application features essential functionalities such as starting competitions, loading results and other data from StarExec or visualizing and filtering such data.

Termination is a branch of theoretical computer science where programs such as algorithms are examined, whether they terminate, that is whether they complete. In terms of StarExec, solvers are running over a bunch of problems returning an answer for each problem: Yes, No or Maybe.

The Termination Competition is an annual event of the termination community of StarExec, where solvers from different contributors compete in several categories. In 2014 the competition was organized by my adviser, Prof. Johannes Waldman, and me. Our goal was to do both, starting the competition run as well as to present it via an automatically updated web-interface.

The 2014 competition caused a total amount of 13 Gigabyte of data and needed about 90 days of CPU time on all of the 192 Quad-Core CPUs of the StarExec cluster.

Our additional goal was to be able to compare these results with those from previous ones. So we added an option to import the data from the 2007 competition hosted by the Laboratory of Computer Science at Université Paris-Sud and the following competitions hosted by the University of Innsbruck.

"StarExec is a cross community logic solving service developed at the University of Iowa (…)." (StarExec 2013) It is a technical infrastructure providing the service to run logic solvers on a powerful cluster of CPUs. It also provides an extensive web-based user interface to upload and run solvers and problems, referred to as benchmarks. A REST-based[1] API is utalized by the Star-Exec-Presenter.

The Star-Exec-Presenter is the software developed as the concrete work for this thesis. It is a web application programmed in Haskel[2] with the utalization of the Yesod Web Framework[3].

The motivation of this thesis is to provide a tool for StarExec users for further research as well as some kind of standardization of hosting and running future termination competition and competitions of other StarExec communities. The goal for Star-Exec-Presenter is to be a tool, that is easy to use as well as easy to install and run. It is open source, so it can be forked und changed to better meet future needs.

With Star-Exec-Presenter we wanted to extend the ability of the StarExec service with filtering of the results and comparing them with older ones from previous competitions. Contributers of solvers should be able to track their progress of development of their respective solver through time.

[1] Representational state transfer, a paradigm to implement a server-client communication
[2] a functional, non-strict, declarative programming language (http://www.haskell.org/)
[3] a REST-based web application framework (http://www.yesodweb.com/)

2
Termination Competition

In this chapter I will discuss the terms of Termination and Termination of Term Rewriting. I also will describe the Termination Competition, as its 2014 incarnation was one of the goals towards the Star-Exec-Presenter was developed.

Termination of Term Rewriting

Termination Community

Termination Competition

3 StarExec

StarExec is the service that is utilized by the Termination Community and other communities to run their research as well as the latest Termination Competition. This chapter will examine the interfaces from which the relevant data is fetched as well as their special characteristics. I will go into the details of how StarExec works and how to get the data.

Interfaces

4
Use Cases

In this chapter I will list the use cases which the Star-Exec-Presenter is mapped onto. These use cases range from simple tasks, for instance displaying solver-results from StarExec, up to more complex ones, like starting and monitoring a competition.

Displaying Results and Outputs

The core of StarExec is not only to run the solver on specific benchmarks but also to host the results and outputs of each run. So a major task for the Star-Exec-Presenter is to display that data in an appropriate way.

Compare Results

Simply displaying the results and outputs of each solver-run is not enough. It is more important to compare that data between several solver.

Interpret Outputs as possible XML-Proofs

Some solver explicitly log an XML-proof with each run in its output. This proof should be detected as well as extracted from the output. Also the respective XSL-File should be used to display the proof in the browser.

Start a predefined Competition

One of the core use cases of the Star-Exec-Presenter is the ability to start a competition run on StarExec. As StarExec actually doesn't know what a competition in our terms is, Star-Exec-Presenter should know it. So all meta-information of a competition should be managed by the Star-Exec-Presenter. These meta information include hierarchical information of the competition's organisation as well as the details which we'll send to or get from StarExec.

Hierarchical information is meant as how each solver and benchmark is organised within the competition. Each competition consists of several meta categories which cover a general topic in the competition. The 2014 Termination Competition has the following: "Termination of Term Rewriting (and Transition Systems)", "Complexity Analysis of Term Rewriting" and "Termination of Programming Languages".

Each meta category is devided into smaller, more specific categories, each being a representation of an actual job on StarExec. These categories may be best described by the meta categories "Termination of Programming Languages", as there is a category about the programming language C. Other categories could bring up Java, Logic Programming or Functional Programming.

Displaying a Competition and its Results

5 Requirements

This chapter discusses the requirements that were agreed to. Which data is important and how is it handled? How should the application be build, which kind of data should be accessible through the Star-Exec-Presenter? And how should the application incorporate with StarExec? All requirements are infered from the previous chapter.

Solver and Post processor output

Caching

Compact web interface

6 Star-Exec-Presenter

The Star-Exec-Presenter is actually the concrete work of this thesis. As a REST-based web application it is an interface to the StarExec service and it provides a caching mechanism. This chapter will give detailed insights in the architectural approach to build the Star-Exec-Presenter according to the requirements.

Application Structure

REST interface

7 Implementation

The Star-Exec-Presenter is a web application written in Haskell and based upon the Yesod Web Framework. In this chapter I will talk about the actual implementation of the tool as well as about the benefits of using especially Haskell and Yesod.

Haskell

Yesod Web Framework

Routes

Templates

Persistence

8 Evaluation

Future considerations

- caching could be automated with a permanent background-worker which traverses the database's content after a given time period
- ajax-based implementation of the web-interface
- generalized export of all data

References

StarExec. 2013. "About StarExec." https://www.starexec.org/starexec/public/about.jsp.