UNIVERSITÄT BAYREUTH

University of Bayreuth

Institute for Computer Science

Bachelor Thesis

in Applied Computer Science

Topic:	Randomly Generated CFGs, The CYK-Algorithm And A GUI
Author:	Andreas Braun <andreasbraun5@aol.com> Matrikel-Nr. 1200197
Version date:	January 17, 2017
1. Supervisor:	Prof. Dr. Wim Martens
2. Supervisor:	Tina Trautner

To my parents.

Abstract

Here will be found the abstract of this thesis.

Zusammenfassung

Hier steht die Zusammenfassung dieser Bachelorarbeit.

Contents

1 Introduction

2 Technology Overview

3 Course of Action

The informal goal is to find a suitable combination of a grammar and a word that meets the demands of an exam exercise. Also the CYK pyramid and one derivation tree of the word must be generated automatically as a "solution picture".
Firstly the exam exercise must have an upper limit of variables per cell while computing the CYK-pyramid.
Secondly the exam exercise must have one or more "special properties" so that it can be checked if the students have clearly understood the algorithm, e.g. "Excluding the possibility of luck."

The more formal goal is identify and determine parameters that in general can be used to define the properties of a grammar, so that the demanded restrictions are met. Also parameters could be identified for words, but "which is less likely to contribute, than the parameters of the grammar." [Second appointment with Martens]

Some introductory stuff:

Possible basic approaches for getting these parameters are the Rejection Sampling method and the "Tina+Wim" method.
Also backtracking plays some role, but right now I don't know where to put it. Backtracking is underapproach to Rejection Sampling.
Note: Starting with one half of a word and one half of a grammar.

Identify restrictions (=parameters) regarding the grammar.
Maybe find restrictions regarding the words, too.

Procedures for automated generation. Each generation procedure considers different restrictions and restriction combinations. The restrictions within one generation procedure can be optimized on its own. Up till now:
Generating grammars: DiceRolling, ...
Generating words: DiceRolling, ...

Parameter optimisation via theoretical and/or benchmarking approach.
Benchmarking = generate N grammars and test them, (N=100000).
Define a success rate and try to increase it.

The overall strategy is as following:

1.) Identify theoretically a restriction/parameter for the grammar. Think about the influence it will have. Think also about correlations between the restrictions.
2.) Validate the theoretical conclusion with the benchmark. Test out the influence of this parameter upon the success rate. Try different parameter settings.

The ordering of step 1 and step 2 can be changed.

Used software

Listings

Following are some interesting classes referenced in the thesis that were too long to fit into the text.

Tables

This section contains all tables referenced in this thesis.

References

[1] JSR 220: Enterprise Java Beans 3.0 `https://jcp.org/en/jsr/detail?id=220`, 09/09/2015