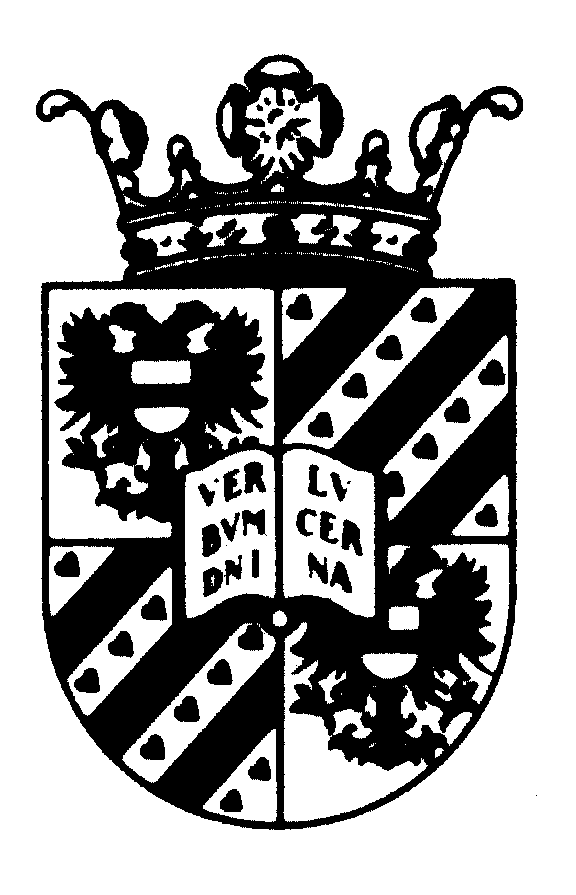
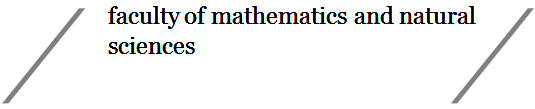
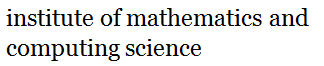
**Final Project Bachelor Computing Science**

**Course code: WBCS13000**

**Credits (EC): 15 points (420 hours)**

**1. General information**

**Project title:** Deriving Syntax Highlighters from Context-Free Grammars

**Date:   
   
First supervisor (name + signature):** Tijs van der Storm

**Second supervisor (name + signature):** Mircea Lungu

**Student (name + signature + student number):** Edser Apperloo

*s2968878*

**2. Project description:**

Language workbenches provide an integrated experience for developing software languages. Language users are supported by editor services, such as syntax highlighting, jump-to-definition, outlines etc. The goal of this project is to leverage existing editors for providing such services. More specifically, the project is about generating syntax highlighting support based on context-free grammars.

Many editors (e.g., VS Code, Textmate, SublimeText, Atom, ACE, CodeMirror etc.) or highlighters (e.g., Highlight.js, Github) accept state-based language definitions for defining syntax highlighting. In the context of the [Rascal](http://www.rascal-mpl.org) language workbench, however, coloring is derived from context-free grammars.

How can we derive state-based highlighters from Rascal's context-free grammars?

**3. Methodology and timeline:**

As a starting point, the student will take the approach detailed in this paper: Mohri, Nederhof, *Regular approximation of context-free grammars through transformation*, Robustness in language and speech technology, 2001, Springer, [[pdf](https://drona.csa.iisc.ernet.in/~deepakd/atc-common/RegApprox.pdf)].

Resources on state-based language tokenizers:

* [TextMate language grammars](https://manual.macromates.com/en/language_grammars)
* [Common JavaScript Syntax Highlighting Specification](https://github.com/mozilla/skywriter/wiki/Common-JavaScript-Syntax-Highlighting-Specification)

Informal milestones

* Provide a detailed description of the abstract syntax of Rascal's grammar formalism and its semantics
* Define the abstract syntax of an intermediate grammar formalism to represent regular approximations of context-free grammars, catering for Rascal's lexical disambiguation constructs (e.g., follow restrictions, keyword reservation, etc.)
* Transform a Rascal grammar to the intermediate representation using the Mohri/Nederhof algorithm.
* Map the resulting regular approximation to the "Textmate" state-based "mode" data type.
* Generate concrete JSON/XML/... for specific editors to be able to test the highlighters.
* Evaluate the approach using various grammars from Rascal's standard library, including Rascal's own grammar, on one or more editors.

**4. Deliverables:**

* A prototype for transforming grammars into highlighters (in Rascal).
* A precise description of the algorithm, including limitations and trade-offs.
* Evaluation of the prototype on several Rascal-defined languages, including Rascal itself.

**5. Grading**

This section should briefly describe how the grade will be composed by summing up weighted contributions on the following five topics:

* **Process:** This assesses the work process throughout the entire period. Typical BSc projects have a 20-25% weight for *process*.
* **Thesis:** This assesses the final BSc thesis manuscript. Typical BSc projects have a 25-30% weight for *thesis.*
* **Technical contributions:** This assesses the software-related deliverables, such as source code, demonstrators, and their documentation and manuals. Typical BSc projects have a weight ranging between 0% and 20% for *technical contributions*.
* **Scientific contribution:** This assesses the research component of the thesis work. Typical BSc projects have a weight around 30% for *scientific contribution*.
* **Final presentation:** This assesses the quality of the final presentation during the BSc symposium. Typical BSc projects have a weight of about 20% for this element.