Parallel parsing in Yi

Project proposal for master's thesis*

Tobias Olausson

gusolaut@student.gu.se

Background

Syntax highlighting is a key feature in modern day text editors. Programmers want to get information about how well their code conforms to the programming language grammar, but without having to compile or run their code. Most text editors use regular expressions to find keywords and highlight other key aspects of the code. While this is fast, it cannot provide any information beyond the lexical syntax of the language.

A nicer approach would be to use the language's own grammar to check against the written code. If the editor had a lexer and parser integrated for the language, it could not only markup the relevant parts of the code, it would also allow the programmer to use any metadata that the parser generates to manipulate the code on an abstract level.

Problem description

Recent research has shown that parsing of programming language text can be done efficiently by using a divide-and-conquer algorithm[1]. Another MSc thesis by Hugo and Hansson has created a way to generate an incremental divide-and-conquer lexer given a context-free grammar[2].

The aim of this project is to combine these two results and implement them into the Yi text editor[3]. The result would be an editor that, in an efficient manner, could give meaningful syntax information given any context-free grammar.

*Aim for specialization in Algorithms, Languages and Logic

Delimitations

The project will be concerned with only the lexer generator and parsing, other parts of the Yi editor will be left untouched as much as possible. Cool applications using the syntax information generated by the lexer and parser will not be a priority, but rather a bonus if time allows for it.

Methodology

This project is mainly concerned with implementing theories already proven into a practical application. Obviously, before implementing anything the existing Yi code base must be carefully examined. When it comes to the actual code, much work will revolve around testing and debugging the code. The resulting parser should be easily checked against existing grammars both in the editor and in a more stand-alone environment.

Project plan

The project is planned for the spring semester of 2014. Implementation and integration of the components will be performed in several steps. The lexer generator obviously has to be in place before the parser can be implemented. Very coarse steps:

1. Study the background papers and Yi source code
2. Integrate the lexer generator into Yi
3. Implement the parallel parser into Yi
4. Use syntactic information for cool stuff (if time permits it)

Supervisor

This project was suggested by Jean-Philippe Bernardy, and as one of the authors of the recent paper on parallel parsing and as one of the main contributors to Yi, he is likely the most suitable for supervising the project.

Prerequisities

This will be a project in computing science at University of Gothenburg. Since the Yi editor and most other software, such as BNFC[4], involved in this project will be written in Haskell, good knowledge in Haskell is required.

Since it also deals with the initial tasks of a compiler, knowledge in programming languages and compilers is also required. Courses that fulfil these requirements include advanced functional programming (DIT260), programming languages (DIT229/230) and compiler construction (DIT300). I have read all of these courses.

References

[1] Efficient Divide-and-Conquer Parsing of Practical Context-Free Languages
Jean-Philippe Bernardy, Koen Claessen
ICFP 2013

[2] A generator of incremental divide-and-conquer lexers
Jonas Hugo, Christoffer Hansson
MSc Thesis, Chalmers University of Technology, draft as of 2013-11-21

[3] Yi: an editor in Haskell for Haskell.
Jean-Philippe Bernardy
In Proc. of the first ACM SIGPLAN symposium on Haskell, pages 61–62. ACM, 2008.

[4] The BNF Converter
http://bnfc.digitalgrammars.com/