Synopsis for Bachelorproject

Regular Expression Matching In Genomic Data

Rasmus Haarslev - nkh877
Troels Thomsen - qvw203

Supervisors: Rasmus Fonseca

23. Februar 2015

Department of Computer Science
University of Copenhagen

1 Problem definition

We wish to determine the possibility of converting sequence analysis patterns used for scan-for-matches[6], into regular expressions[3] and test their efficiency against the KMC[5] engine.

Specifically we wish to solve the following problems:

- Is it possible to programatically convert patterns used by the scan-for-matches program into regular expressions for the KMC engine? If not all patterns used by scan-for-matches then which ones?
- Is it possible to achieve speeds matching or exceeding scan-for-matches with the generated regular expressions and the KMC engine?
- Can we find weak extensions to regular expressions, which would enable us to support more or all scan-for-matches patterns?

1.1 Limits

- We will not attempt to modify the KMC engine.
- We will not look into the theory behind the scan-for-matches engine.

2 Motivation

Institute for Microbiology have generated, and are still generating, a lot of DNA sequencing data. Currently they have around a total of 2 petabytes of data. These sequences of DNA contain a lot of information, but searching through the data currently uses the scan-for-matches program, which while performing very well, is not very user friendly and have some unfortunate limitations when running many consecutive scans, since it performs I/O operations for every run.

Recently, Professor Henglein's group has developed a regular expression engine called KMC, which so far have performed five times better than current industry standard engines. Since scan-for-matches outperforms NR-grep[1], we are hoping that optimized regular expressions running on KMC will be able to outperform NR-grep and subsequently scan-for-matches.

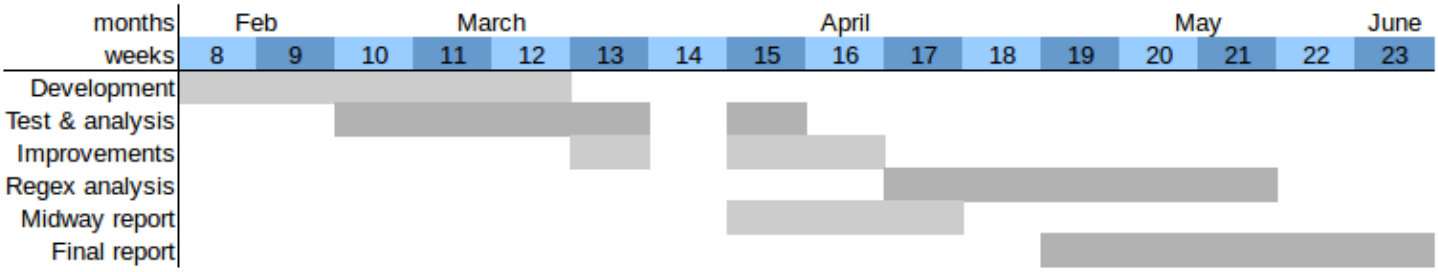
If we could achieve a performance improvement over scan-for-matches, it would greatly benefit the bioinformatics team. As such we see this as a chance to make a unique contribution to ongoing and future research projects, while at the same time providing a chance for the KMC team to have their engine tested in a new scenario.

3 Tasks and Schedule

- Develop a standalone Ruby and C application as a solution to the tasks defined in the problem definition.
  - **Product:** A fully functional Ruby/C application, that can translate scan-for-matches patterns into regular expressions, understood by the KMC engine.
  - **Resource demands:** Laptops, our contact persons with insight in the KMC engine, testing data in the fasta format, and the KMC engine.
  - **Dependencies:** Problem definition
  - **Time demands:** 5 weeks
- Test and analyze the efficiency of our regular expressions on the KMC engine, compared to scan-for-matches.
  - **Product:** An extensive analyzis of our regular expressions in combination with the KMC engine, with suggestions for possible improvements.
  - **Resource demands:** Laptops, testing data in the fasta format, and the KMC engine.
  - **Dependencies:** The application that we developed.
  - **Time demands:** 5 weeks (In parallel with development)
- Improve upon our application based on our tests and analysis.
  - **Product:** An improved application for translating scan-for-matches patterns into regular expressions, understood by the KMC engine.
  - **Resource demands:** Laptops, our application, testing data in the fasta format, and the KMC engine.
  - **Dependencies:** The written analysis.
  - **Time demands:** 3 weeks
- Analyze our regular expressions, and look for possible extensions, which would enable more or all scan-for-matches patterns.
  - **Product:** An extension to our regular expressions.

- **Resource demands:** Laptops.
- **Dependencies:** Our translated regular expressions.
- **Time demands:** 4 weeks

3.1 Gantt Diagram



References

[1] Gonzalo Navarro. Nr-grep: A fast and flexible pattern matching tool. http://www.dcc.uchile.cl/~gnavarro/ps/spe01.pdf. Visited 18th February 2015.

[2] Ulrik Terp Rasmussen Niels Bjørn Bugge Grathwohl, Fritz Henglein. Two-pass greedy regular expression parsing. *Implementation and Application of Automata, volume 7982 of Lecture Notes in Computer Science*, pages 60–71, 2013.

[3] Ulrik Terp Rasmussen Niels Bjørn Bugge Grathwohl, Fritz Henglein. A crash-course in regular expression parsing and regular expressions as types. *Department of Computer Science (DIKU), University of Copenhagen*, pages 1–37, 2014.

[4] Ulrik Terp Rasmussen Niels Bjørn Bugge Grathwohl, Fritz Henglein. Optimally streaming greedy regular expression parsing. *Theoretical Aspects of Computing — ICTAC 2014*, pages 224–240, 2014.

[5] KMC Team. Kleene meets church: Regular expressions and types. http://www.diku.dk/kmc/. Visited 18th February 2015.

[6] The SEED Team. Scan for matches. http://blog.theseed.org/servers/2010/07/scan-for-matches.html. Visited 18th February 2015.