Midtvejsrapport 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

Contents

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?

2 Translation of scan-for-matches patterns into regular expressions

2.1 Mismatches, Insertions and Deletions

The subpatterns can have the following form, which allows for mismatches, insertions and deletions in the subpattern.

$$x_1 \dots x_n[m,i,d] \quad \{x_1 \dots x_n \in A \mid m,i,d \in \mathbb{N}_0\} \tag{1}$$

This notation allows for 0 or the given number of mismatches, insertions or deletionsin, or all possible combinations in between. This means we for an example can have m mismatches and i insertions, but not necessarily d deletions.

2.1.1 Mismatches and Deletions

We can combine the matching of mismatches and deletions to somewhat simplify our expressions. We can express one mismatch and one deletion in the following regular expression.

$$((x_1?|.)\ x_2 \cdots x_n) \mid (x_1?\ .\ x_3 \cdots x_n) \mid \quad \cdots \quad \mid (x_1? \cdots x_{n-1}\ .) \mid \tag{2}$$

$$(.\ x_2? \cdots x_n) \mid (x_1\ (x_2?|.)\ x_3 \cdots x_n) \mid \quad \cdots \quad \mid (x_1\ x_2? \cdots x_{n-1}\ .) \mid \tag{3}$$

$$\vdots$$

$$(.\ x_2 \cdots x_n?) \mid (x_1\ .\ x_3 \cdots x_n?) \mid \quad \cdots \quad \mid (x_1 \cdots x_{n-1}\ (x_n?|.)) \tag{4}$$

This quite large expression quickly grows into an even larger expression, if we have multiple mismatches or deletions. $\hookrightarrow$ denotes the continuation of the current line.

$$\begin{aligned}
&((((x_1?|.)\ x_2 \cdots x_n) \mid ((x_1?|.)(x_2?|.) \cdots x_n) \mid \quad \cdots \quad \mid ((x_1?|.)(x_2?|.) \cdots (x_n?|.))) \mid \quad (5)\\
\hookrightarrow\ &(((x_1?|.)\ x_2 \cdots x_n) \mid ((x_1?|.)\ x_2\ (x_3?|.) \cdots x_n) \mid \quad \cdots \quad \mid ((x_1?|.)\ x_2\ (x_3?|.) \cdots (x_n?|.))) \mid\\
&\hookrightarrow \quad \vdots\\
&\hookrightarrow (((x_1?|.)\ x_2 \cdots x_n) \mid \quad \cdots \quad \mid ((x_1?|.)\ x_2 \cdots x_{n-1}\ (x_n?|.)))) \mid
\end{aligned}$$

$$\begin{aligned}
&(x_1((((x_2?|.) \cdots x_n) \mid ((x_2?|.)\ (x_3?|.) \cdots x_n) \mid \quad \cdots \quad \mid ((x_2?|.)\ (x_3?|.) \cdots (x_n?|.))) \mid \quad (6)\\
\hookrightarrow\ &(((x_2?|.) \cdots x_n) \mid ((x_2?|.)\ x_3\ (x_4|.) \cdots x_n) \mid \quad \cdots \quad \mid ((x_2?|.)\ x_3\ (x_4?|.) \cdots (x_n?|.))) \mid\\
&\hookrightarrow \quad \vdots\\
&\hookrightarrow (((x_2?|.) \cdots x_n) \mid \quad \cdots \quad \mid ((x_2?|.)\ x_3 \cdots x_{n-1}\ (x_n?|.))))) \mid\\
&\qquad\quad \vdots\\
&(x_1 \quad \cdots \quad x_{n-1}\ (x_n?|.)) \qquad\qquad (7)
\end{aligned}$$

2.1.2 Insertions

2.1.3 Backreferencing

One way to simplify our expression, is to use backreferences. With backreferences we can express m mismatches, d deletions and i insertions in the following regular expressions.

2.1.3.1 Mismatches

$$((x_1?|.) \mid (x_2?|.) \mid \cdots \mid (x_n?|.)) \qquad (8)$$

2.1.3.2 Deletions

2.1.3.3 Insertions

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.