Support tools in the specification and lexical analysis phases: Haskell

Authorship: Joel Aumedes - Marc Cervera

Language processors - Maria Teresa Alsinet Bernadó

Degree in computer engineering

Polytechnic school - University of Lleida

Index of contents

Abstract

This presentation is going to be about programming languages that support regular expressions, concretely we're going to focus on Haskell and his defining module regular expressions, Regex.

First of all we've to understand that Haskell is:

- ❖ A programming language developed by the programming languages research community.
- ❖ Is a lazy, purely functional language.
    - ➢ Purely functional programming usually designates programming paradigm that treats all the computation as the evaluation of mathematical functions.
- ❖ Born as an open source vehicle for programming language research.
- ❖ Particularly useful for programs that manipulate data structures and for concurrent/parallel programming.

Programming language that supports regular expressions, Haskell

Regex, what is it?

“Regex”, is the contraction of “Regular expression”.

A regex, is a sequence of characters that specifies a search pattern in a text.

Normally, that patterns are used for “find” or “find and replace” operations on strings.

It is a technique developed in theoretical computer science and formal language theory.

Programming language that supports regular expressions, Haskell

Regex, what is it?

To match regular expressions in Haskell we use the module Text.Regex.

This module is defined as Text.Regex.Base and is implemented in multiple backends.

We are going to use the backend Text.Regex.Posix, that is the Unix module.

Programming language that supports regular expressions, Haskell

Regex in Haskell, overview

Regex Syntax in Haskell is the one we have seen in class, with:

- Ranges [a-zA-Z]
- Repetitions *, +, {2,5}
- Disjunction (“yes”|”no”)
- Case sensitive

With the addition of anchors:

- \` Beginning of the text
- \’ End of the text
- \< Beginning of the word
- \> End of the word

Regex Examples

“sun”	Matches the string “sun” in the text, separated or not. For example, “sunflower” would also match.
“\\`Dear readers”	Matches the text if it begins with “Dear readers”
“\\`[a-z]+\\’”	Matches when the whole text is lowercase, at least one letter and no spaces or other characters
“\\<[1-9][0-9]{0,3}\\>”	Matches numbers from 1 to 9999, if they are a word. For example, it would not match “mcr007”
“\\<([a-z]+|[A-Z]+)\\>”	Matches words that are either all lowercase or all uppercase

How to test for Regex matches

The Text.Regex modules implement two infix operators to test for matches, since they don’t implement find-and-replace.

- =~ :: Text String -> Regex String -> Target
- =~~ :: Text String -> Regex String -> Monad Target

The Target can be either Bool or String.

Examples:

- “aaabba” =~ “ab” :: Bool
  - True
- “Good Luck” =~ “L[a-z]+” :: String
  - “Lu”

Regex in Haskell, overview

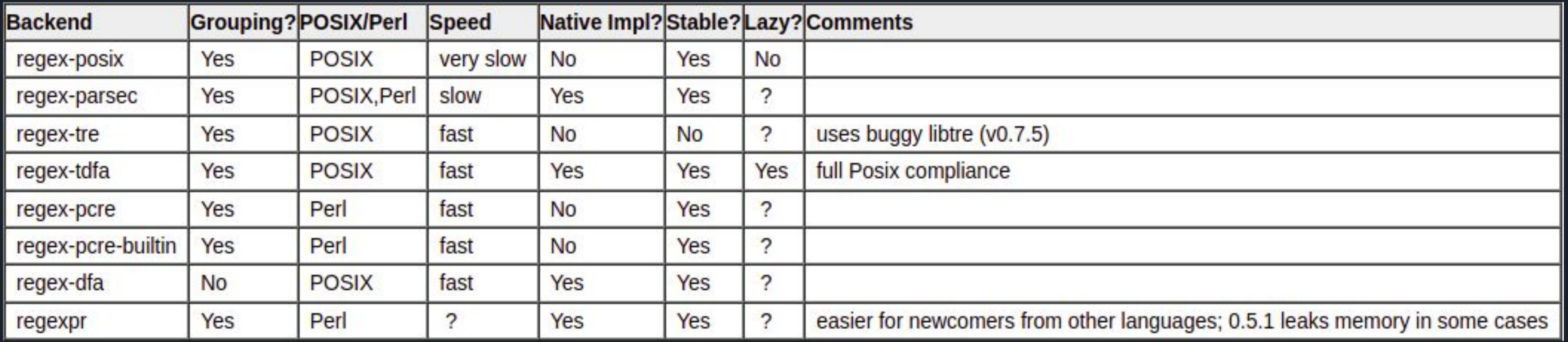
Kris Klukewicz developed a regex library for Haskell that has been implemented with a variety of backends.

Some of these backends are native Haskell implementations, others are not and rely on external C libraries such libpcre.

There are also a number of alternate or complementary regex libraries, including:

- Bryan O’Sullivan’s => text-icu
- Yoshikuni Jujo’s => regexpr
- Don Stewart’s => pcre-light
- Martin Sulzmann’s => regexpr-symbolic
- Matt Morrow’s => regexqq
- Uwe Schmidt’s => hxt-regex-xmlschema

Regex in Haskell, overview



Backend	Grouping?	POSIX/Perl	Speed	Native Impl?	Stable?	Lazy?	Comments
regex-posix	Yes	POSIX	very slow	No	Yes	No	
regex-parsec	Yes	POSIX,Perl	slow	Yes	Yes	?	
regex-tre	Yes	POSIX	fast	No	No	?	uses buggy libtre (v0.7.5)
regex-tdfa	Yes	POSIX	fast	Yes	Yes	Yes	full Posix compliance
regex-pcre	Yes	Perl	fast	No	Yes	?	
regex-pcre-builtin	Yes	Perl	fast	No	Yes	?	
regex-dfa	No	POSIX	fast	Yes	Yes	?	
regexpr	Yes	Perl	?	Yes	Yes	?	easier for newcomers from other languages; 0.5.1 leaks memory in some cases

We've focused on regex-posix because we've done all the tests in a Linux system. The second reason to have used regex-posix, is because is the most beginner-friendly.

Bibliography

- https://wiki.haskell.org/Regular_expressions#Sample_benchmark
- https://www.haskell.org/alex/doc/html/regexps.html
- https://en.wikipedia.org/wiki/Regular_expression
- https://en.wikipedia.org/wiki/Haskell_(programming_language)
- https://en.wikipedia.org/wiki/Purely_functional_programming
- http://www.serpentine.com/blog/2007/02/27/a-haskell-regular-expression-tutorial/