Applicative regular expressions

Roman Cheplyaka

April 20, 2012
Dutch HUG Day

Intro

Two uses of regular expressions:

- Recognition
  - grep
  - search in text editors
  - lexer generators
- Parsing

Capture groups

Example: a regex for URLs

```
(http|ftp)://(([^./]+)\.)+([^./]+)(/([^/]+))+/?
```

Combinators

A better approach: parsing combinators

Regular languages	Applicative functors
Language concatenation	`<*>`
Language union	`<|>`
Kleene star	`many`

Example

```
data Protocol = HTTP | FTP
data URL = URL Protocol [String] [String]

url :: RE Char URL
url = URL <$> protocol <* string "://" <*> host <*> path

protocol :: RE Char Protocol
host, path :: RE Char [String]
protocol = HTTP <$ string "http"
       <|> FTP  <$ string "ftp"
host = (:) <$> s <*> some (sym '.' *> s)
    where s = some $ psym $ not . (`elem` ['.','/'])
path = some (sym '/' *> s)
    where s = some $ psym $ not . (== '/')
```

Comparison

Comparison with monadic parsing libraries:

- better complexity
- better memory usage
- incremental parsing
- longest match

Perl vs POSIX

Perl vs POSIX semantics

- Perl for inner matches
- POSIX or Perl for the whole match

Challenges

Type safety vs speed

Current state

- Available on hackage and github
- Feature-full
- Performance needs some improvement

Where we want to get

Dominate the world

... of regex libraries for Haskell