Parsec Intro

Oleg Grenrus

April 22, 2015

Parse

(Computing) Analyse (a string or text) into logical syntactic components:

a user question input is parsed into an internal conceptual representation

Shallow dive into the theory

Chomsky hierarchy

TODO: IMAGE

Wikipedia

Type	Name / Formalism / Recongizing automaton	Example
0	Recursively enumerable unrestricted grammar Turing machine	
1	Context-sensitive context sensitive grammar Linear-bounded non-deterministic Turing machine	$a^n b^n c^n, n \in \mathbb{N}$
2	Context-free context free grammar Non-deterministic pushdown automaton	$a^n b^n, n \in \mathbb{N}$
3	Regular regular expressions Finite state automaton	$a^n, n \in \mathbb{N}$ $a^n b^m, n$ and m are even

There are different languages (i.e. types of input), and different parsing methods (i.e. libraries)

	regular	applicative	monadic	ad-hoc
	`regex-applicative`	`parsec`		–
regular	⊤	⊤	⊤	⊤
context free		⊤	⊤	⊤
context sensitive			⊤	⊤

Regular language

a^n, $n \in \mathbb{N}$.

Easy to identify with PCRE regex:

```
import Text.Regex.PCRE

regex :: Regex
regex = makeRegex "^a*$"

regexTest :: String -> Bool
regexTest = matchTest regex
```

Regular language – Ad-hoc

Also easy to identify using ad-hoc approach:

```
adhocTest :: String -> Bool
adhocTest = all (== 'a')
```

Another regular language

$a^n b^m$, n and m are even.

Home exercise: Try to write a regular expression.

Naive ad-hoc approach

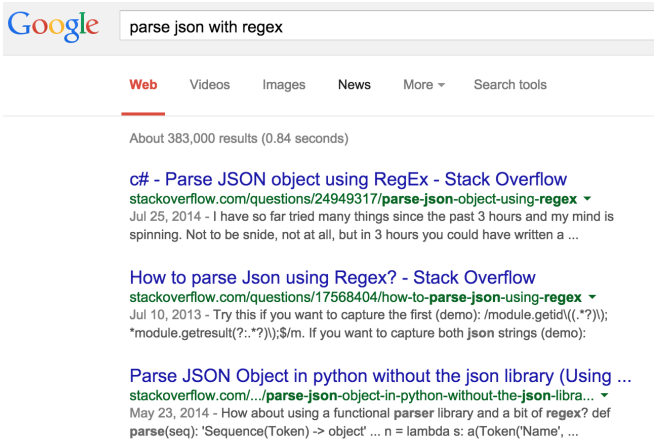
```
naiveAdhocTest :: String -> Bool
naiveAdhocTest input = onlyAandB && even countA && even countB
  where onlyAandB = all (\c -> c == 'a' || c == 'b') input
        countA    = length (filter (== 'a') input)
        countB    = length (filter (== 'b') input)
```

Another regular language – Regex + Ad-hoc

We can write `onlyAandB` using regular expressions abstraction:

```
regexAdhocTest :: String -> Bool
regexAdhocTest input = onlyAandB && even countA && even countB
  where onlyAandB = matchTest (makeRegex "^[ab]*$" :: Regex) input
        countA    = length (filter (== 'a') input)
        countB    = length (filter (== 'b') input)
```

Interlude


Google
parse json with regex
Web
Videos
Images
News
More
Search tools
About 383,000 results (0.84 seconds)
c# - Parse JSON object using RegEx - Stack Overflow
stackoverflow.com/questions/24949317/parse-json-object-using-regex
Jul 25, 2014 - I have so far tried many things since the past 3 hours and my mind is spinning. Not to be snide, not at all, but in 3 hours you could have written a ...
How to parse Json using Regex? - Stack Overflow
stackoverflow.com/questions/17568404/how-to-parse-json-using-regex
Jul 10, 2013 - Try this if you want to capture the first (demo): /module.getid\((.*?)\); *module.getresult(?:.*?)\);$/m. If you want to capture both json strings (demo):
Parse JSON Object in python without the json library (Using ...
stackoverflow.com/.../parse-json-object-in-python-without-the-json-libra...
May 23, 2014 - How about using a functional parser library and a bit of regex? def parse(seq): 'Sequence(Token) -> object' ... n = lambda s: a(Token('Name', ...

Parsing JSON

. . . or almost JSON:

```
{
  key1: 10,                          // number
  key2: [ true, false, null ],       // array of bools and nulls
  key3: [ foo, bar, { bar: foo } ]  // identifiers and recursion
}
```

1 Grammar definition

$$
\begin{aligned}
\mathit{scalar} &\leftarrow \texttt{null} \mid \mathit{boolean} \mid \mathit{number} \mid \mathit{ident} \\
\mathit{boolean} &\leftarrow \texttt{true} \mid \texttt{false} \\
\mathit{number} &\leftarrow [0 \ldots 9]^{+} \\
\mathit{ident} &\leftarrow [a \ldots z]^{+} \\
\mathit{array} &\leftarrow [\,] \mid [\ \mathit{value}\ (\texttt{,}\ \mathit{value})^{*}\ ] \\
\mathit{object} &\leftarrow \{\} \mid \{\ \mathit{pair}\ (\texttt{,}\ \mathit{pair})^{*}\ \} \\
\mathit{pair} &\leftarrow \mathit{ident}\ \texttt{:}\ \mathit{value} \\
\mathit{value} &\leftarrow \mathit{scalar} \mid \mathit{array} \mid \mathit{object}
\end{aligned}
$$

Data definition

```
data Value = AJNull
           | AJBool Bool
           | AJNumber Integer
           | AJText String
           | AJList [Value]
           | AJObject (Map String Value)
```

And now let's live code the rest.