

# 問題記述と形式化 (中井担当分) レポート課題

2017 年度担当: 中井央

2017.12.18

## 課題内容

### 課題にあたっての注意事項

本レポート課題では、作成したプログラムとそれを説明した文書の双方を提出する必要がある。プログラム作成についての注意事項を以下に述べる。

以下の項目を遵守すること。

- プログラムは Ruby で作成すること。
- プログラム作成にあたっては、この課題全体のディレクトリ (フォルダ) の名前を各自の学籍番号とすること。ディレクトリ名には半角数字を用いること。
- 全体ディレクトリの中に課題番号をディレクトリ名とするサブディレクトリを設置すること。
- 各課題番号ディレクトリ内にはその課題のために必要なファイルのみ作成すること。また、`Readme.txt` というファイルを作成し、このディレクトリ内のファイルの一覧を含め、各ファイルの簡単な説明 (例: 構文解析器クラスが定義されたファイル) を含めておくこと。`Readme.txt` の中には、それ以外に必要なことがらがあれば、それらを含めてよい。
- 各ファイル名は、指定があるものはその指示に従うこと。
- 課題を満たしていることを確認するためのデータファイルや出力結果が納められたファイルは適宜、当該ディレクトリへ含めてよい。

## 課題の概要

JSON というデータ形式がある。以下のステップを踏んで、課題のプログラムを作成せよ。作成は講義で述べた方法に基づいて行うこと。 各ステップの詳細は後述する。各ステップの配点は次の通りである。中井担当分を 100 点満点で採点し、最終評価は、鈴木先生担当分と中井担当分を同一比率にした後の合計点にて行う。

1. JSON を入力として受け付ける構文解析器の作成 (40 点)
2. 構文解析木を作らずに構文解析木を表示するための dot 形式の出力をする構文解析器の作成 (20 点)
3. 構文解析木を作成し、各ノードクラスにてアクションを定義し、構文解析木を表現する dot 形式を出力するプログラムの作成 (20 点)
4. 構文解析木を作成し、Visitor を用いることで、構文解析木を表現する dot 形式を出力するプログラムの作成 (20 点)

いずれも動作しない中途半端なプログラムの場合は採点の対象としない。

以下、それぞれの詳細を記す。

## JSON の構文解析器の作成

この講義で述べた方法に基づいて JSON 形式の入力を受け付ける構文解析器を作成せよ。JSON の文法を図 1 に与える。字句解析クラスのプログラムを図 3

与えておく。適宜変更して使用してよい。JSON のサンプルを図 2 に示す。JSON そのものについては各自で調べられたい。

```
json : object
      | array ;

object : LBRA members RBRA ;

members :
        | memberlist ;

memberlist : pair (COMMA pair)* ;

pair : STRING COLON value ;

array : LBRACKET elements RBRACKET ;

elements :
          | elementlist ;

elementlist : value (COMMA value)* ;

value : STRING
        | INT
        | FLOAT
        | TRUE
        | FALSE
        | NULL
        | object
        | array
        ;
```

図 1: JSON の文法

```
{
  "Fred" : "Flintstone",
  "Wilma" : 1234,
  "Barney" : false,
  "Dino" : [ "Tyrannical Dog", 3.14159, true ],
  "Bam Bam" :
  {
    "strength" : 515,
    "size" : "small",
    "Pebble" : { "lastname" : "Flintstone",
                  "male" : false}
  }
}
```

図 2: example0.json

式の出力については、第 2 回講義メモ (2) で示した `rdp2graph.rb` を参考にすること。なお、文字列や値は、それがわかるようにノード内に表示できるようにすること。出力例を図 4 に示す。

## ツリーの作成とその上での操作

課題 3 および 4 について述べる。

どちらの課題もまず、ツリーを作る必要がある。ノードの作成にあたっては、各文法記号がノード種別となるようにクラスを作成すること。

## アクションによる dot 形式の出力

ここではアクションによって、dot 形式の出力をして、構文解析木を得られるようにすること。dot 形

<pre> class JsonLexer   def initialize(f)     @srcf=f     @line=""     @linenumber=0   end    attr_reader :linenumber    def lex()     if /\s+/ =~ @line       @line = \$'     end     while @line.empty? do       @line = @srcf.gets       if @line == nil         return :eof       end       @linenumber += 1       if /\A\s+/ =~ @line         @line = \$'       end     end   end end </pre>	<pre>     case @line     when /\A\: /       yield \$&amp;       token = :colon     when /\A\[ /       yield \$&amp;       token = :lbracket     when /\A\] /       yield \$&amp;       token = :rbracket     when /\A\{/       yield \$&amp;       token = :lbra     when /\A\}/       yield \$&amp;       token = :rbra     when /\A\, /       yield \$&amp;       token = :comma     when /\A\[~\]"*\"/       yield \$&amp;       token = :string     when /\A[+-]?\d+\.\d+([eE][+-]?[0-9]+)?/       yield \$&amp;       token = :float     when /\A[+-]?\d+/       yield \$&amp;       token = :int     when /\Afalse/       yield \$&amp;       token = :false     when /\Atrue/       yield \$&amp;       token = :true     when /\Anull/       yield \$&amp;       token = :null     when /\A\s/       # ignore       token = :whitespace     when /\A\S/       # ignore       token = :other     end     @line = \$'     puts "matched is #{token}(#{ \$&amp;})"     return token   end end </pre>
---	---

图 3: json\_lexer.rb

