PEG意見交換会 (2007/11/25)

PEG Example Generator (略してPEG)

稲葉一浩 (k.inaba)
http://www.kmonos.net/wlog/

発端

① WikipediaのPEGのページで勉強中

② “The PEG Mailing List” なるものを発見

③ ログ読み開始

④ 一番最近のスレッドが…

発端

トーマスさん

俺 APG っていうパーザジェネレータ作ってるんだけど、

これと PEG ってもしかしてめちゃめちゃ似てない？？

発端

APGってよく知らないけど
　A ← ‘a’ A ‘a’　/　‘a’
って文法が「‘a’が奇数個並んだ文字列」じゃなくて「‘a’が2^n-1個並んだ文字列」にマッチするなら、PEGぽいね

シュミッツさん

つまり

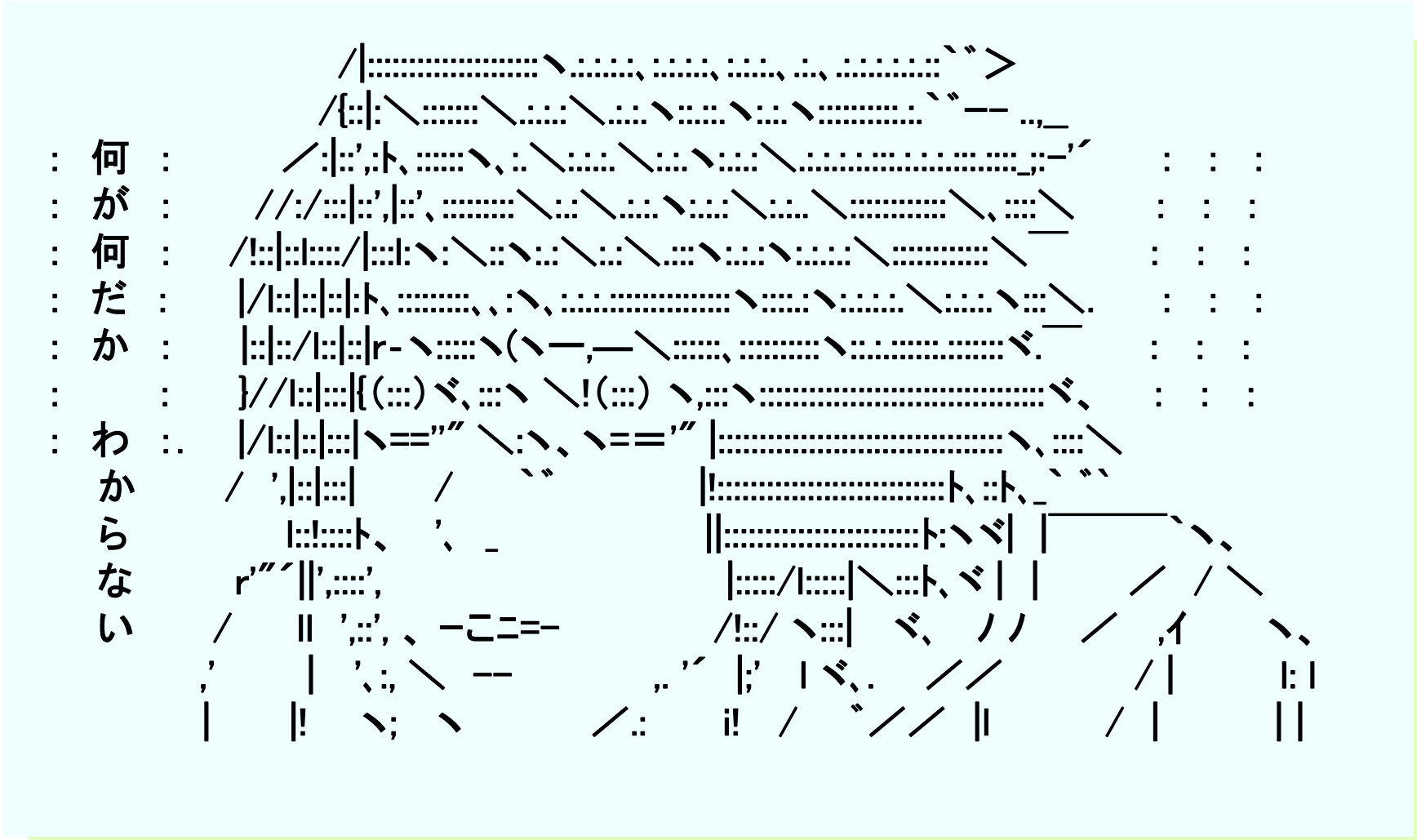
- PEG では

  A ← ‘a’ A ‘a’　/　‘a’

  は

  ”aaaaa”

  の全体にマッチしないのだ！


何
が
何
だ
か
わ
わか
らな
い

発端

- 何が何だかわからなかったので
- PEGを見たらどういう文字列にマッチするかわかるようになるために

PEGの文法定義を入れたら
マッチする文字列を列挙してくれる
プログラムを書きました

参考文献

- 正規表現からその正規表現にマッチするような例を生成する (Ruby)
  - http://d.hatena.ne.jp/soutaro/20070516/1179323606
- 正規表現にマッチする例 (JavaScript)
  - http://www.kmonos.net/wlog/73.html#_1543070517
- 大人げ (OCaml)
  - http://d.hatena.ne.jp/sumii/20070517/p2

おことわり

- 理論的に
  - 「PEGにマッチする例を完全に列挙」は不可能
  - なので
    - 全部例を列挙し終わったあと無限ループしたり
    - ちょっと列挙漏れがあったり

  するのは仕様と言うことで

論より証拠 :: デモ

- “abab...” にマッチするPEG

```
ab = Pegexp.new <<PEG
  A <- a B / a
  B <- b A / b
PEG

ab.example.take(10)
```

論より証拠 :: デモ

- ‘a’ が 2^n-1 個並んだ文字列

```
a = Pegexp.new <<PEG
  A <- a A a / a
PEG

a.example.take(5)
```

論より証拠 :: デモ

- $a^n b^n c^n$ (n>0) にマッチするPEG
  - PredicateもRepeatも未対応なので苦労しました

```
abc = Pegexp.new <<PEG
  S <- I A Y
   I<- J TZ /
   J<- K TZ /
   K<- X bZ / X
   Z<- TZ/
   T<- a/b/c
  A <- aA  / a
  X <- aXb / ab
  Y <- bYc / bc
PEG

abc.example {|s| puts s}
```

論

Pegexp#match

- 文字列がPEGにマッチするか判定する
  - 要するに普通のPEGパーザ
  - 正規表現の例生成なら正規表現マッチエンジンを実装する必要はなかったけれど、PEGの場合はたぶん必須（後述）
- 実装：PEGをRubyのPackrat Parserに変換
  - 実装は40行くらい
  - PEG の / がプログラミング言語の || に直接変換できるのでかなり楽

Pegexp#example

- 例を生成する enumerator

試行錯誤...

- 文字列をランダム生成
  - マッチする物だけ選んで返す
    - すごく遅い。無茶すぎる
- PEG を CFG と思って例を生成
  - マッチする物だけ選んで返す
    - 今回の実装はPredicate未対応につき全部Choiceに変換しているため、無駄が酷い。
    - Predicate 対応した場合は悪くない選択と思われる

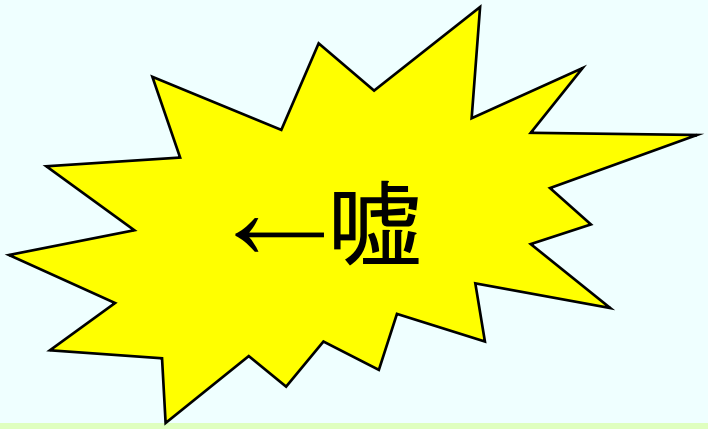
できるだけ直接生成

- example(e1 / e2 / e3) =
  - example(e1)
  - + (example(e2) - e1にマッチするもの)
  - + (example(e3) - e1やe2にマッチするもの)

#match
が
必要

- example(e1 e2) =

```
for s1 in  example(e1)
  for s2 in example(e2)
    s1 + s2
```



できるだけ直接生成

後ろから順に
生成してみる
アルゴリズム！

- example(e1 e2) =
  for s1 in example(e1)
    for s2 in example(e2)
      s1 + s2 の内 e1.match(s1+s2)==s1.size な物

- example(e1 e2) =
    for s2 in example(e2)
      for s1 in example(e1) のうち後ろにs2が来ても
        s1 + s2　　食っちゃわないもの

こんな感じの関数で実装

- def generate(e, um, follow)
  - eにマッチする文字列のうち
  - 後ろにfollowが来てもそれを食っちゃわないで
  - しかも後ろにfollowが来てもumとマッチしない
  - ようなものを生成しまくる関数

end

面倒だった点1

- 右再帰
  - PEGの文法なら左再帰は無いはずだけど
  - 右再帰は普通にある
  - 右から生成アルゴリズムには鬼門
  - とりあえずループチェックで右再帰検出してます

面倒だった点2

- ダメな例無限生成しまくるループ
  - S ← C A B
  - A ← aA / a　　# a+
  - B ← bBc / bc　# b^n c^n
  - C ← 略　　# a^n b^n になってる

とりあえず
100回連続マッチ失敗
したら
強制ループ脱出

  - for s1 in [“bc”, “bbcc”, “bbbccc”, …]
    for s2 in [“a”, “aa”, “aaa”, …]　　←ここで無限ループ
      for s3 in [“”]
        s3+s2+s1 が a^n b^n 先読み成功したら
        yield s3+s2+s1

面倒だった点3

- スタックオーバーフロー
- Packrat　Parserを再帰で書いてるので
  入力文字長段の再帰になる可能性がある
- メモ化のための結果記憶処理が最後に
  入るので、どーやっても末尾再帰にならない
  （CPS変換する？めんどい？？）