テキスト処理 第7回 (2008-06-03)
try拡張レポート解説

田中哲
産業技術総合研究所
情報技術研究部門
akr@isc.senshu-u.ac.jp
http://staff.aist.go.jp/tanaka-akira/textprocess-2008/

レポート

- 選択(:alt) を任意個引数に拡張せよ
- 連接(:cat) を任意個引数に拡張せよ

- 〆切 2008-06-03 12:00
- RENANDI
- 拡張子が txt なプレインテキストが望ましい

[:alt, r1, r2, r3, ...]の動作例

- try([:alt, "a", "b", "c"], %w[a], 0) {|pos| p pos }
  #=> 1
- try([:alt, "a", "b", "c"], %w[b], 0) {|pos| p pos }
  #=> 1
- try([:alt, "a", "b", "c"], %w[c], 0) {|pos| p pos }
  #=> 1
- try([:alt, "a", [:cat, "a", "a"]], %w[a a], 0) {|pos|
    p pos
  }
  #=> 1, 2

拡張された try_alt

- 渡された引数それぞれについて try を呼び出すだけ

```
def try_alt(re, str, pos)
  1.upto(re.length-1) {|i|
    try(re[i], str, pos) {|pos2| yield pos2 }
  }
end
```

[:cat, r1, r2, r3, ...]の動作例

- try([:cat, "a", "b", "c"], %w[a b c], 0) {|e| p e }
  #=> 3
- try([:cat, [:alt, "a", [:cat, "a", "a"]],
  [:alt, "a", [:cat, "a", "a"]],
  [:alt, "a", [:cat, "a", "a"]]],
  %w[a a a a a a], 0) {|pos| p pos }
  #=> 3,4,4,5,4,5,5,6

:cat の任意個引数のヒント

- 再帰を使う
- 難しいかも?
- 出来なかった場合、3引数、4引数を作ってみよ
  - [:cat3, r1, r2, r3]
  - [:cat4, r1, r2, r3, r4]

拡張された try_cat

```
def try_cat(re, str, pos)
  if re.length == 1
    yield pos
  else
    re2 = [:cat]
    2.upto(re.length-1) {|i| re2 << re[i] }
    try(re[1], str, pos) {|pos2|
      try(re2, str, pos2) {|pos3| yield pos3 }
    }
  end
end
```

try_cat の中身 (1)

- 再帰で実装

```
def try_cat(re, str, pos)
  if re.length == 1
    yield pos
  else
    re2 = [:cat]
    2.upto(re.length-1) {|i| re2 << re[i] }
    try(re[1], str, pos) {|pos2|
      try(re2, str, pos2) {|pos3| yield pos3 }
    }
  end
end
```

try_cat の中身 (2)

- [:cat] なら再帰しない
- そのときは [:empstr] と同じ

```
def try_cat(re, str, pos)
  if re.length == 1
    yield pos
  else
    re2 = [:cat]
    2.upto(re.length-1) {|i| re2 << re[i] }
    try(re[1], str, pos) {|pos2|
      try(re2, str, pos2) {|pos3| yield pos3 }
    }
  end
end
```

try_cat の中身 (3)

- [:cat, r1, ...] なら2つに分ける
- r1 r1 は re[1]
- [:cat, ...] re2 に作る
- 再帰的に分けていけばそのうち空になって再帰は止まる

```
def try_cat(re, str, pos)
  if re.length == 1
    yield pos
  else
    re2 = [:cat]
    2.upto(re.length-1) {|i| re2 << re[i] }
    try(re[1], str, pos) {|pos2|
      try(re2, str, pos2) {|pos3| yield pos3 }
    }
  end
end
```

3引数限定連接

- :cat3 のところを try_cat じゃなくて try_cat3 を呼び出すように変える
- あるいは、:cat のところで 3引数の場合に try_cat3 を呼び出すように変えてもいい

```
when :cat3
  raise ArgumentError, "unexpected number of argum
  try_cat3(re, str, pos) {|pos2| yield pos2 }
```

try_cat3 の定義

```
def try_cat3(re, str, pos)
  try(re[1], str, pos) {|pos2|
    try(re[2], str, pos2) {|pos3|
      try(re[3], str, pos3) {|pos4|
        yield pos4
      }
    }
  }
end
```

try_cat よりネストが一段深い

4引数限定連接

- :cat4 のところは :cat3 と同様

```
def try_cat4(re, str, pos)
  try(re[1], str, pos) {|pos2|
    try(re[2], str, pos2) {|pos3|
      try(re[3], str, pos3) {|pos4|
        try(re[4], str, pos4) {|pos5|
          yield pos5
        }
      }
    }
  }
end
```

ネストがさらに一段深い

再帰とネスト

- 再帰を使うと、任意段のネストを実現できる
- 任意段のネストはループでは困難
- 任意個引数の連接はループでは困難

ざっと見た結果

- alt の拡張はだいたい出来ている
- cat の任意個引数はやはり難しかった
  - できたひとも存在した
- 3, 4 引数の cat はもうすこし出来たひとが多い
  - 4,5人?
- catもループで拡張しようとして失敗しているひとが多い
  - 再帰を使うと書いておいたのに
- rtf で提出したひとがいたがプレインテキストにしてほしい