Challenge

Spare Time Teaching

April 11, 2015

Introduction

People have told us that you can't do something big in the Lambda Calculus, but we don't agree.

Problem

Regular expressions are very basic and important in Computer Science. Unfortunately only very few students have actually written a regular expression matcher, so here is the chance.

Write, in the Lambda Calculus, a regular expression matcher without using the Y combinator.

Use the datatype:

```
regexp ::= Empty
         | Epsilon
         | Atom nat
         | Concat regexp regexp
         | Altern regexp regexp
         | Kleene regexp
```

```
mk_empty   ≡ λ v_Em v_Ep f_At f_C f_Al f_K . v_Em
mk_epsilon ≡ λ v_Em v_Ep f_At f_C f_Al f_K . v_Ep
mk_atom    ≡ λ n .
  λ v_Em v_Ep f_At f_C f_Al f_K .
    f_At n
mk_concat  ≡ λ r_1 r_2 .
  λ v_Em v_Ep f_At f_C f_Al f_K .
    f_C (r_1 v_Em v_Ep f_At f_C f_Al f_K)
        (r_2 v_Em v_Ep f_At f_C f_Al f_K)
mk_altern  ≡ λ r_1 r_2 .
  λ v_Em v_Ep f_At f_C f_Al f_K .
    f_Al (r_1 v_Em v_Ep f_At f_C f_Al f_K)
         (r_2 v_Em v_Ep f_At f_C f_Al f_K)
mk_kleene  ≡ λ r .
```

```
  λ v_Em v_Ep f_At f_C f_Al f_K .
    f_K (r v_Em v_Ep f_At f_C f_Al f_K)


match (2⋆)  [2; 2] →* T
match (2⋆⋆) [2; 2; 1] →* F
match ( (00 | 000)1 ) [0; 0; 0; 1] →* T
```