RaML 15150 Tests Summary

Yinglan Chen

April 11, 2020

1 HW1

Implement factorial function as below.

```
let rec fact x =
  if x <= 1 then 1
  else let _ = Raml.tick 1.0 in x * fact (x - 1)
let _ = fact 5
```

The rest cannot be implemented since RaML does not support real numbers.

2 HW2

Implement all functions, but none of the function can be analyzed. For example, here is the implementation of `add`

```
let rec add ((n : int), (m: int)) = if n = 0 then m
  else let _ = Raml.tick 1.0 in 1 + add (n-1, m)
```

The error message shows"A bound for {function name} could not be derived. The linear program is infeasible"
The reason is RaML does not support integer recursion (because of negative int issues). A better error message can be: `integer type in recursion detected. Please use rnat instead`
Luckily, this assignment does not use any negative integer. So I change int to rnat type and reimplement HW2
Here is the `add` function after re-implementation

```
let rec add (n, m) =
    Rnat.ifz n
        ((fun () -> m))
        ((fun n' -> let () = Raml.tick 1.0 in succ(add (n', m)) ))

let _ = add (succ(succ(Rnat.zero)),Rnat.zero);;
```

Function `pascal` is still infeasible.

```
let rec pascal ((i: Rnat.t), (j: Rnat.t)) =
    Rnat.ifz j
        ((fun () -> succ(Rnat.zero)))
        ((fun i' ->
            ( let (i_minus_j,_) = (Rnat.minus i j) in
              (Rnat.ifz i_minus_j
              ((fun () -> succ(Rnat.zero)))
              ((fun j' -> let _ = Raml.tick 2.0 in
                          Rnat.add (pascal (i',j')) (pascal(i', j)))))
            )
        ))

let _ = pascal (succ(succ(Rnat.zero)),succ(Rnat.zero));;
```

The reason is pascal has exponential cost. A suggested error message is "The linear program is infeasible. function name has exponential cost". A suggested implementation (during weekly meeting) is when the program sees two recursive calls in one branch, replace one occurrence of the recursive call with a constant and analyze, repeat the same thing on the other occurrence. If both analysis are linear, that implies the function has exponential cost, but I don't know how to achieve that exactly.

There is no natural number comparison in Rnat module. So I use `Rnat.to_int` to implement `n < d` in function `div_mod`

```
exception DivideByZero
let rec divmod (n,d) =
  Rnat.ifz d
  ((fun () -> raise DivideByZero))
  ((fun d' ->
    ( Rnat.ifz n
      ((fun () -> (Rnat.zero, Rnat.zero)))
      ((fun n' ->
        if (Rnat.to_int n) < (Rnat.to_int d) then (Rnat.zero, n)
        else  let (diff, _) = Rnat.minus n' d' in
              let (x,y) = divmod(diff, d) in (Rnat.succ x, y)
      ))
    )
  ))
```

When analyzing `is_prime`, RaML works fine when given degree 1, the tight upper bound, but has Uncaught exception when given any degree larger than 1. The error message is shown below

```
Resource Aware ML, Version 1.4.1, July 2018

Typechecking expression ...
```

```
  Typecheck successful.
  Stack-based typecheck successful.

Analyzing expression ...

  Trying degree: 3Uncaught exception:

  Not_found

Raised at file "src/map.ml" (inlined), line 428, characters 6-26
Called from file "src/map.ml", line 1273, characters 23-77
Called from file "raml/annotations.ml", line 703, characters 11-33
Called from file "list.ml", line 111, characters 24-34
Called from file "list.ml", line 111, characters 24-34
Called from file "list.ml", line 111, characters 24-34
Called from file "raml/indices.ml", line 392, characters 17-44
Called from file "list.ml", line 137, characters 24-31
Called from file "src/list0.ml" (inlined), line 27, characters 40-75
Called from file "src/list.ml", line 161, characters 2-19
Called from file "raml/analysis.ml", line 1924, characters 24-70
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1982, characters 25-192
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1790, characters 25-148
Called from file "raml/analysis.ml", line 1790, characters 25-148
Called from file "raml/analysis.ml", line 1908, characters 25-193
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1908, characters 25-193
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1982, characters 25-192
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 917, characters 30-42
Called from file "raml/analysis.ml", line 1259, characters 31-64
Called from file "raml/analysis.ml", line 1377, characters 34-55
Called from file "raml/analysis.ml", line 1291, characters 38-62
Called from file "raml/annotations.ml", line 577, characters 19-42
Called from file "raml/analysis.ml", line 1529, characters 21-54
Called from file "raml/analysis.ml", line 1637, characters 33-48
Called from file "raml/analysis.ml", line 1829, characters 16-231
Called from file "raml/analysis.ml", line 1845, characters 32-53
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1908, characters 25-193
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1982, characters 25-192
Called from file "raml/analysis.ml", line 1672, characters 18-153
```

```
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1790, characters 25-148
Called from file "raml/analysis.ml", line 1790, characters 25-148
Called from file "raml/analysis.ml", line 1908, characters 25-193
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1908, characters 25-193
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1982, characters 25-192
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 917, characters 30-42
Called from file "raml/analysis.ml", line 1259, characters 31-64
Called from file "raml/analysis.ml", line 1377, characters 34-55
Called from file "raml/annotations.ml", line 577, characters 19-42
Called from file "raml/analysis.ml", line 1529, characters 21-54
Called from file "raml/analysis.ml", line 1637, characters 33-48
Called from file "raml/analysis.ml", line 1829, characters 16-231
Called from file "raml/analysis.ml", line 1845, characters 32-53
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1908, characters 25-193
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1908, characters 25-193
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1908, characters 25-193
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 917, characters 30-42
Called from file "raml/annotations.ml", line 577, characters 19-42
Called from file "raml/analysis.ml", line 1529, characters 21-54
Called from file "raml/analysis.ml", line 1637, characters 33-48
Called from file "raml/analysis.ml", line 1829, characters 16-231
Called from file "raml/analysis.ml", line 1845, characters 32-53
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1672, characters 18-153
Called from file "raml/analysis.ml", line 1760, characters 14-133
Called from file "raml/analysis.ml", line 2081, characters 16-28
Called from file "main.ml", line 578, characters 8-21
```

3 HW3

Cannot find a good way to test in RaML. It does not support `assert` in ocaml. Functions infeasible:

`filterInt, look_and_say, look_say_table, subset_sum, subset_sum_cert`

`subset_sum` and `subset_sum_cert` are exponential function, thus the output is expected. As stated in HW2, a suggested error message is to point out the exponential cost.

The reason why `filterInt`, `look_and_say`, and `look_say_table` are unknown. I suspect it is because they use the result of a recursive helper function as an argument. Here is the code of `filterInt`. If I replace the last line with filter(x,r), Raml can give a resource bound.

```
let rec filterInt((x:int),(l:int list)) =
  match l with
  | [] -> let _ = Raml.tick 1.0 in []
  | a::r -> if heads(x,l) = 0 then let _ = Raml.tick 1.0 in a :: filterInt(x,r)
        else
            let _ = Raml.tick 1.0 in let tail = tails(x,l) in
            filterInt(x, tail)
```

4 HW4

type supports `int * tree * tree` but not `tree * int * tree`, so following code has to change according to that.
Anaylze mode works normally.

5 HW5

cannot support poly type since RaML does not support real.
`'a forest` type, defined as `Node of 'a option * 'a forest list` ,is also not supported because the size of `'a forest list` cannot be determined.