Homework 4 Solutions

98-317: Hype for Types

Due: 13 February 2018 at 11:59 PM

1 Required

Consider the following syntax:

```
Sort   Name       Operators
Var    x    ::=   x
Num    n    ::=   Zero
                  One
                  Plus(n, n)
Fun    f    ::=   Fun(x, n)
App    a    ::=   App(f, n)
```

This syntax is quite restrictive. For example, Plus can only have subtrees that are of sort Num - any ASTs of sort Fun, App, or Var are not allowed. Similarly, Fun's second subtree must be either Zero, One, or Plus(n, n). It may not be an AST of sort Var, Fun, or App.

Some examples of programs that use this syntax are as follows:

```
App(Fun(x, Zero), Zero)
Plus(Plus(Plus(Zero, One), One), One)
Fun(x, Plus(Plus(Zero, One), One))
```

Req Task 1 In SML, we can define the same kinds of expressions:

Our Syntax	SML
`x`	`x`
`Zero`	`0`
`One`	`1`
`Plus(n, n)`	`n + n`
`Fun(x, n)`	`fn x => n`
`App(f, n)`	`f n`

but SML allows us to write many more programs than this system does. Give an example of a program that you can write using the SML expressions written above that would not be allowed in our syntax.

Solution `Fun(x, x)`, since x has sort Var, not sort Num.

Req Task 2 What is the arity of `App`?

Solution `(Fun, Num) App`

Req Task 3 Draw the abstract syntax tree for `App(Fun(x, Zero), Zero)`. What is the sort of `App(Fun(x, Zero), Zero)`?

Solution

```
       App
     /     \
  Fun      Zero
 /    \
x    Zero
```

Sort: App