Nano2 Reference

Expression syntax:

```
e ::= n | x | e1 + e2 | let x = e1 in e2 | \x -> e | e1 e2
```

Operational semantics:

```
                e1  =>  e1'
[Add-L]   --------------------
          e1 + e2  =>  e1' + e2

                e2  =>  e2'
[Add-R]   --------------------
          n1 + e2  =>  n1 + e2'

[Add]     n1 + n2  =>  n      where n == n1 + n2

                       e1  =>  e1'
[Let-Def] ---------------------------------------
          let x = e1 in e2  =>  let x = e1' in e2

[Let]     let x = v in e2  =>  e2[x := v]

             e1  =>  e1'
[App-L]   -----------------
          e1 e2  =>  e1' e2

            e  =>  e'
[App-R]   -------------
          v e  =>  v e'

[App]     (\x -> e) v  =>  e[x := v]
```

Syntax of types:

```
T ::= Int | T1 -> T2 | a
S ::= T | forall a . S
```

Typing rules:

```
[T-Num]  G |- n :: Int

         G |- e1 :: Int   G |- e2 :: Int
[T-Add]  ------------------------------
               G |- e1 + e2 :: Int

[T-Var]  G |- x :: S          if x:S in G

           G, x:T1 |- e :: T2
[T-Abs]  ----------------------
         G |- \x -> e :: T1 -> T2

         G |- e1 :: T1 -> T2   G |- e2 :: T1
[T-App]  ---------------------------------
                  G |- e1 e2 :: T2

         G |- e1 :: S   G, x:S |- e2 :: T
[T-Let]  -------------------------------
            G |- let x = e1 in e2 :: T

         G |- e :: forall a . S
[T-Inst] ---------------------
          G |- e :: [a / T] S

               G |- e :: S
[T-Gen]  ---------------------- if not (a in FTV(G))
         G |- e :: forall a . S
```

Here $n \in \mathbb{N}$ is natural number, $v \in$ Val is a value, $x \in$ Var is a variable, $e \in$ Expr is an expression, $a \in$ TVar is a type variable, $T \in$ Type is a mono-type, $S \in$ Poly is a poly-type, $G \in$ Var $\to$ Poly is a type environment (a context).