ids: 209300581, 316097575

Typing rules:

```
Typing rule set!:
  For every: type environment _Tenv,
             variable reference _x1
             expressions _e1 and
             type expressions _S1:
  If _Tenv |- _e1 : _S1 and
     _Tenv |- _x1 : _S1
  Then _Tenv |- (set! _x1 _e1) : void
```

```
Typing rule LitExp:
  For every: type environment _Tenv,
             symbol expression _sy
             compound sexp _sexp
             number expression _num
             boolean expression _bol
             string expession _str
  _Tenv |- _sy: Symbol(_sy)
  _Tenv |- '(): Symbol
  _Tenv |- _sexp: Pair
  _Tenv |- _num: Number
  _Tenv |- _bool: Boolean
  _Tenv |- _str: String
```

```
 Typing rule type-case:
For all user-defined-type id
        with component records record_1 ... record_n
        with fields (field_ij) (i in [1...n], j in [1..R_i])
        val CExp
        body_i for i in [1..n] sequences of CExp
  if _Tenv |- val : T0
     _Tenv |- record_i: T0 for i in [1..n]
     _Tenv |-  body_i for i in [1..n] : T1

     then _Tenv |- ( type-case id val (record_1 (field_11 ... field_1r1)
body_1)...  ): T1
```

```
 Typing rule define-type:
     for all user-defined-type expression:
     _Tenv |- (exp : DefineTypeExp) : void
```

1.

a. False. g accepts type **T1**, but **a** is a number. There's no guarantee that **T1** is a number hence g(a) might not be defined.
b. False. f accepts type **T2** and **x** is **T1** hence f(x) might not be defined.
c. True. f accepts type **T1** and returns **T2**. The lambda wrapping f simply returns the value of that f returns, **x** is of type **T1** hence f(x) is of type **T2** so the lambda will return type **T2** as well
d. True. f accepts T1 X T2 variables and gets **x** and **y**. **y** is of type **T2** and **x** can be inferred because he is bound variable. The lambda returns the value of f which is **T3**

Question 2 in the next page

| question 2a | ((lambda (f x1) (f 1 x1)) + #t) |
|---|---|

Rename bound variables: ((lambda (f x1) (f 1 x1)) + #t)

**Assign type variables:**

| expression | variables |
|---|---|
| ((lambda (f x1) (f 1 x1)) + #t) | T0 |
| (lambda (f x) (f 1 x)) | T1 |
| (f 1 x) | T2 |
| f | Tf |
| 1 | Tnum1 |
| x | Tx |
| + | T+ |
| #t | T#t |

**Construct type equations:**

| expression | equation |
|---|---|
| ((lambda (f x1) (f 1 x1)) + #t) | T1 = [T+ * T#t -->T0] |
| (lambda (f x) (f 1 x)) | T1 = [Tf * Tx --> T2] |
| (f 1 x) | Tf = [Tnum1 * Tx --> T2] |
| 1 | Tnum1 = Number |
| + | T+ = [Number * Number --> Number] |
| #t | T#t = Boolean |

**Solving equations:**

| | equation | substitution |
|---|---|---|
| 1 | T1 = [T+ * T#t -->T0] | |
| 2 | T1 = [Tf * Tx --> T2] | |
| 3 | Tf = [Tnum1 * Tx --> T2] | |
| 4 | Tnum1 = Number | |
| 5 | T+ = [Number * Number --> Number] | |
| 6 | T#t = Boolean | |

(T1 = [T+ * T#t --> T0]) ○ Substitution = (T1 = [T+ * T#t --> T0])

| | equation | substitution |
|---|---|---|
| 2 | T1 = [Tf * Tx --> T2] | { T1 := [T+ * T#t --> T0] } |
| 3 | Tf = [Tnum1 * Tx --> T2] | |
| 4 | Tnum1 = Number | |
| 5 | T+ = [Number * Number --> Number] | |
| 6 | T#t = Boolean | |

(T1 = [Tf * Tnum1 * Tx -->]) ○ Substitution = ([T+ * T#t --> T0] = [Tf * Tx --> T2])

| | equation | substitution |
|---|---|---|
| 3 | Tf = [Tnum1 * Tx --> T2] | { T1 := [T+ * T#t --> T0] } |
| 4 | Tnum1 = Number | |
| 5 | T+ = [Number * Number --> Number] | |
| 6 | T#t = Boolean | |
| 7 | Tf = T+ | |
| 8 | Tx = T#t | |
| 9 | T2 = T0 | |

(Tf = [Tnum1 * Tx -->T2]) ○ Substitution = Substitution ○ (Tf = [Tnum1 * Tx ->T2])

| | equation | substitution |
|---|---|---|
| 4 | Tnum1 = Number | { T1 := [T+ * T#t --> T0], Tf := [Tnum1 * Tx --> T2] } |
| 5 | T+ = [Number * Number --> Number] | |
| 6 | T#t = Boolean | |
| 7 | Tf = T+ | |
| 8 | Tx = T#t | |
| 9 | T2 = T0 | |

(Tnum1 = Number) ○ Substitution = Substitution ○ (Tnum1 = Number)

| | equation | substitution |
|---|---|---|
| 5 | T+ = [Number * Number --> Number] | { T1 := [T+ * T#t --> T0], Tf := [Number * Tx --> T2], Tnum1 = Number } |
| 6 | T#t = Boolean | |
| 7 | Tf = T+ | |
| 8 | Tx = T#t | |
| 9 | T2 = T0 | |

(T+ = [Number * Number --> Number]) ○ Substitution = Substitution ○ (T+ = [Number * Number -> Number])

| | equation | substitution |
|---|---|---|
| 6 | T#t = Boolean | { T1 := [ [Number * Number --> Number] * T#t --> T0], Tf := [Number * Tx --> T2], Tnum1 = Number , T+ = [Number * Number -> Number]} |
| 7 | Tf = T+ | |
| 8 | Tx = T#t | |
| 9 | T2 = T0 | |

(T#t = Boolean ) ○ Substitution = Substitution ○ (T#t = Boolean )

| | equation | substitution |
|---|---|---|
| 7 | Tf = T+ | { T1 := [ [Number * Number --> Number] * Boolean --> T0], Tf := [Number * Tx --> T2],T+ = [Number * Number -> Number] , T#t = Boolean } |
| 8 | Tx = T#t | |
| 9 | T2 = T0 | |

(Tf = T+) ○ Substitution = Substitution ○ (Tf = T+)

| | equation | substitution |
|---|---|---|
| 8 | Tx = T#t | { T1 := [ [Number * Number --> Number] * Boolean --> T0], Tf := [Number * Tx --> T2],T+ = [Number * Number -> Number], T#t = Boolean } |
| 9 | T2 = T0 | |
| 10 | Tx = Number | |
| 11 | T2 = Number | |

(Tx = T#t) ○ Substitution = Substitution ○ (Tx = T#t)

| | equation | substitution |
|---|---|---|
| 9 | T2 = T0 | { T1 := [ [Number * Number --> Number] * Boolean --> T0], Tf := [Number * T#t --> T2],T+ = [Number * Number -> Number], T#t = Boolean, Tx =T#t } |
| 10 | Tx = Number | |

| | | |
|---|---|---|
| 11 | T2 = Number | |

(T2 = T0) ○ Substitution = Substitution ○ (T2 = T0)

| | equation | substitution |
|---|---|---|
| 10 | Tx = Number | { T1 := [ [Number * Number --> Number] * Boolean --> T0], Tf := [Number * T#t --> T0],T+ = [Number * Number -> Number], T#t = Boolean, Tx = T#t} |
| 11 | T2 = Number | |

(Tx = Number) ○ Substitution = Substitution ○ (Tx = Number)

**now we got Tx= Boolean and Tx = Number so we can say that the expression is not well-typed**

| question 2b | ((lambda (f1 x1) (f x1 1)) + *) |
|---|---|

Rename bound variables: ((lambda (f x) (f x 1)) + *)

**Assign type variables:**

| expression | variables |
|---|---|
| ((lambda (f x) (f x 1)) + *) | T0 |
| (lambda (f x) (f x 1) ) | T1 |
| (f x 1) | T2 |
| f | Tf |
| x | Tx |
| 1 | Tnum1 |
| + | T+ |
| * | T* |

**Construct type equations:**

| expression | equation |
|---|---|
| ((lambda (f x) (f x 1)) + *) | T1 = [T+ * T* -->T0] |
| (lambda (f x) (f x 1)) | T1 = [Tf * Tx --> T2] |
| (f x 1) | Tf = [Tx * Tnum1 --> T2] |
| 1 | Tnum1 = Number |
| + | T+ = [Number * Number --> Number] |
| * | T* = [Number*Number-> Number] |

**Solving equations:**

| | equation | substitution |
|---|---|---|
| 1 | T1 = [T+ * T* -->T0] | |

| | | |
|---|---|---|
| 2 | T1 = [Tf * Tx --> T2] | |
| 3 | Tf = [Tx * Tnum1 --> T2] | |
| 4 | Tnum1 = Number | |
| 5 | T+ = [Number * Number --> Number] | |
| 6 | T* = [Number*Number-> Number] | |

(T1 = [T+ * T* -->T0]) ○ Substitution = (T1 = [T+ * T* -->T0]])

| | **equation** | **substitution** |
|---|---|---|
| 2 | T1 = [Tf * Tx --> T2] | { T1 := [T+ * T*--> T0] } |
| 3 | Tf = [Tx * Tnum1 --> T2] | |
| 4 | Tnum1 = Number | |
| 5 | T+ = [Number * Number --> Number] | |
| 6 | T* = [Number*Number-> Number] | |

(T1 = [Tf * Tx --> T2]) ○ Substitution = (T1 := [T+ * T* --> T0] = T1 = [Tf * Tx --> T2])

\

| | **equation** | **substitution** |
|---|---|---|
| 3 | Tf = [Tx * Tnum1 --> T2] | { T1 := [T+ * T* --> T0] } |
| 4 | Tnum1 = Number | |
| 5 | T+ = [Number * Number --> Number] | |
| 6 | T* = [Number*Number-> Number] | |
| 7 | Tf = T+ | |
| 8 | Tx = T* | |
| 9 | T2 = T0 | |

(Tf = [Tx * Tnum1 --> T2]) ○ Substitution = Substitution ○ (Tf = [Tx * Tnum1 --> T2])

| | **equation** | **substitution** |
|---|---|---|
| 4 | Tnum1 = Number | { T1 := [T+ * T* --> T0], Tf = [Tx * Tnum1 --> T2] } |
| 5 | T+ = [Number * Number --> Number] | |
| 6 | T* = [Number*Number-> Number] | |
| 7 | Tf = T+ | |
| 8 | Tx = T* | |
| 9 | T2 = T0 | |

(Tnum1 = Number) ○ Substitution = Substitution ○ (Tnum1 = Number)

| | **equation** | **substitution** |
|---|---|---|

| | | |
|---|---|---|
| 5 | T+ = [Number * Number --> Number] | { T1 := [T+ * T* --> T0], Tf = [Tx * Number --> T2] , Tnum1=Number} |
| 6 | T* = [Number*Number-> Number] | |
| 7 | Tf = T+ | |
| 8 | Tx = T* | |
| 9 | T2 = T0 | |

= [Number * Number --> Number]) ○ Substitution = Substitution ○ (T+ = [Number * Number -> Number])

| | **equation** | **substitution** |
|---|---|---|
| 6 | T* = [Number*Number-> Number] | { T1 := [[Number*Number->Number] * T*--> T0], Tf = [Tx * Number --> T2] , Tnum1=Number, T+ = [Number * Number -> Number]} |
| 7 | Tf = T+ | |
| 8 | Tx = T* | |
| 9 | T2 = T0 | |

* = [Number*Number-> Number] ) ○ Substitution = Substitution ○ (T* = [Number*Number-> Number] )

| | **equation** | **substitution** |
|---|---|---|
| 7 | Tf = T+ | { T1 := [ [Number * Number --> Number] * [Number * Number --> Number] --> T0], Tf := [Number * Tx --> T2],T+ = [Number * Number -> Number] , T* = [Number*Number-> Number] } |
| 8 | Tx = T* | |
| 9 | T2 = T0 | |

(Tf = T+) ○ Substitution = Substitution ○ (Tf = T+)

| | **equation** | **substitution** |
|---|---|---|
| 8 | Tx = T* | { T1 := [ [Number * Number --> Number] * [Number * Number --> Number] --> T0], Tf := [Number * Tx --> T2],T+ = [Number * Number -> Number] , T* = [Number*Number-> Number], Tf=T+ } |
| 9 | T2 = T0 | |
| 10 | Tx = Number | |
| 11 | T2 = Number | |

(Tx = T*) ○ Substitution = Substitution ○ (Tx = T*)

| | equation | substitution |
|---|---|---|
| 9 | T2 = T0 | { T1 := [ [Number * Number --> Number] * Boolean --> T0], Tf := [Number *[Number*Number->Number]--> T2],T+ = [Number * Number -> Number], [Number*Number-> Number],Tf=T+, Tx =T* } |
| 10 | Tx = Number | |
| 11 | T2 = Number | |

(T2 = T0) ○ Substitution = Substitution ○ (T2 = T0)

| | equation | substitution |
|---|---|---|
| 10 | Tx = Number | { T1 := [ [Number * Number --> Number] * Boolean --> T0], Tf := [Number *[Number*Number->Number]--> T0],T+ = [Number * Number -> Number], [Number*Number-> Number],Tf=T+, Tx =T* } |
| 11 | T2 = Number | |

(Tx = Number) ○ Substitution = Substitution ○ (Tx = Number)

**now we got Tx= [Number*Number->Number] and Tx = Number so we can say that the expression is not well-typed**