

TFAE Grammar

```
<TFAE> ::= <num>
         | {+ <TFAE> <TFAE>}
         | {- <TFAE> <TFAE>}
         | <id>
         | {fun {<id> : <TE>} <TFAE>}
         | {<TFAE> <TFAE>}

<TE> ::= num
       | (<TE> -> <TE>)
```

TFAE Types

```
(define-type Type
  [numT]
  [arrowT (arg : Type)
           (result : Type)])
```

```
(define-type TypeEnv
  [mtEnv]
  [aBind (name : symbol)
         (type : Type)
         (rest : TypeEnv)])
```

TFAE Expressions

```
(define-type TFAE
  [num (n : number)]
  [add (l : TFAE)
       (r : TFAE)]
  [sub (l : TFAE)
       (r : TFAE)]
  [id (name : symbol)]
  [fun (name : symbol)
       (t : Type)
       (body : TFAE)]
  [app (rator : TFAE)
       (rand : TFAE)])
```

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      ...)))
```

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [num (n) ...])))
```

$\Gamma \vdash \langle \text{num} \rangle : \text{num}$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [num (n) (numT)])))
```

$\Gamma \vdash \langle \text{num} \rangle : \text{num}$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [add (l r)
        ... (typecheck l env) ...
        ... (typecheck r env) ...])))
```

$$\frac{\Gamma \vdash \mathbf{e}_1 : \mathit{num} \quad \Gamma \vdash \mathbf{e}_2 : \mathit{num}}{\Gamma \vdash \{+ \mathbf{e}_1 \ \mathbf{e}_2\} : \mathit{num}}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [add (l r)
        (type-case Type (typecheck l env)
          [numT ()
            ... (typecheck r env) ...]
          [else (error 'typecheck
            "add expects a num")]])]))))
```

$$\frac{\Gamma \vdash \mathbf{e}_1 : \mathit{num} \quad \Gamma \vdash \mathbf{e}_2 : \mathit{num}}{\Gamma \vdash \{+ \mathbf{e}_1 \ \mathbf{e}_2\} : \mathit{num}}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [add (l r)
        (type-case Type (typecheck l env)
          [numT ()
            (type-case Type (typecheck r env)
              [numT () (numT)]
              [else (error 'typecheck
                           "add expects a num")])]
          [else (error 'typecheck
                       "add expects a num")])])
      [else (error 'typecheck
                   "add expects a num")]))))
```

$$\frac{\Gamma \vdash e_1 : num \quad \Gamma \vdash e_2 : num}{\Gamma \vdash \{+ e_1 e_2\} : num}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [id (name) ...])))
```

$$[\dots \langle id \rangle \leftarrow \tau \dots] \vdash \langle id \rangle : \tau$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [id (name) (get-type name env)])))
```

$$[\dots \langle id \rangle \leftarrow \tau \dots] \vdash \langle id \rangle : \tau$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [fun (name arg-type body)
        ...])))
```

$$\frac{\Gamma[\langle \text{id} \rangle \leftarrow \tau_1] \vdash \mathbf{e} : \tau_2}{\Gamma \vdash \{\text{fun } \{\langle \text{id} \rangle : \tau_1\} \mathbf{e}\} : (\tau_1 \rightarrow \tau_2)}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [fun (name arg-type body)
        ... (typecheck body (aBind name
                                     arg-type
                                     env)) ... ])))
```

$$\frac{\Gamma[\langle \text{id} \rangle \leftarrow \tau_1] \vdash \mathbf{e} : \tau_2}{\Gamma \vdash \{ \text{fun } \{ \langle \text{id} \rangle : \tau_1 \} \mathbf{e} \} : (\tau_1 \rightarrow \tau_2)}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [fun (name arg-type body)
        (arrowT arg-type
          (typecheck body (aBind name
                                arg-type
                                env))))]))
```

$$\frac{\Gamma[\langle \text{id} \rangle \leftarrow \tau_1] \vdash \mathbf{e} : \tau_2}{\Gamma \vdash \{\text{fun } \{\langle \text{id} \rangle : \tau_1\} \mathbf{e}\} : (\tau_1 \rightarrow \tau_2)}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [app (fn arg)
           ...])))
```

$$\frac{\Gamma \vdash \mathbf{e}_1 : (\tau_2 \rightarrow \tau_3) \quad \Gamma \vdash \mathbf{e}_2 : \tau_2}{\Gamma \vdash \{\mathbf{e}_1 \ \mathbf{e}_2\} : \tau_3}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [app (fn arg)
        ... (typecheck fn env) ...
        ... (typecheck arg env) ...])))
```

$$\frac{\Gamma \vdash \mathbf{e}_1 : (\tau_2 \rightarrow \tau_3) \quad \Gamma \vdash \mathbf{e}_2 : \tau_2}{\Gamma \vdash \{\mathbf{e}_1 \ \mathbf{e}_2\} : \tau_3}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [app (fn arg)
        (type-case Type (typecheck fn env)
          [arrowT (arg-type result-type)
            ... (typecheck arg env) ...]
          [else (error 'typecheck
            "app expects a function")]])))))
```

$$\frac{\Gamma \vdash \mathbf{e}_1 : (\tau_2 \rightarrow \tau_3) \quad \Gamma \vdash \mathbf{e}_2 : \tau_2}{\Gamma \vdash \{\mathbf{e}_1 \ \mathbf{e}_2\} : \tau_3}$$

TFAE Type Checker

```
(define typecheck : (TFAE TypeEnv -> Type)
  (lambda (fae env)
    (type-case TFAE fae
      [app (fn arg)
          (type-case Type (typecheck fn env)
            [arrowT (arg-type result-type)
                  (if (equal? arg-type
                               (typecheck arg env))
                      result-type
                      (error 'typecheck
                            "arg mismatch")))]
            [else (error 'typecheck
                        "app expects a function")])])]))
```

$$\frac{\Gamma \vdash \mathbf{e}_1 : (\tau_2 \rightarrow \tau_3) \quad \Gamma \vdash \mathbf{e}_2 : \tau_2}{\Gamma \vdash \{\mathbf{e}_1 \ \mathbf{e}_2\} : \tau_3}$$

TFAE Type Checker

```
(define get-type : (symbol TypeEnv -> Type)
  (lambda (name env)
    (type-case TypeEnv env
      [mtEnv () (error 'typecheck
                       "unbound identifier")]
      [aBind (name2 type rest)
              (if (equal? name name2)
                  type
                  (get-type name rest))]))))
```