Assignment 1

Evaluation Contexts, Typing Rules, Run-Time Semantic Rules, and Type Soundness

Nick Zuber

March 1, 2018

1 Complete Language Grammar

Types

$\langle T \rangle$ ::= `Bool`
| `Int`
| $\langle T \rangle \to \langle T \rangle$
| $\langle T \rangle \times \langle T \rangle$
| `List` $\langle T \rangle$

Expressions

$\langle e \rangle$ ::= `true`
| `false`
| 0
| `for` $\langle e \rangle$ `then` $\langle e \rangle$ `else` $\langle e \rangle$
| `succ` $\langle e \rangle$
| `pred` $\langle e \rangle$
| `isZero` $\langle e \rangle$
| $\langle x \rangle$
| $\lambda\langle x \rangle.\langle e \rangle$
| $\langle e \rangle\ \langle e \rangle$
| $<\langle e \rangle, \langle e \rangle>$
| `fst` $\langle e \rangle$
| `snd` $\langle e \rangle$
| `nil`
| `cons` $\langle e \rangle\ \langle e \rangle$
| `head` $\langle e \rangle$
| `tail` $\langle e \rangle$
| `isNil` $\langle e \rangle$
| $\langle er \rangle$

Errors

$\langle er \rangle$ ::= `error`

Values

$\langle v \rangle$ ::= `true`
| `false`
| 0
| `succ` $\langle v \rangle$
| $\lambda\langle x \rangle.\langle e \rangle$
| $<\langle v \rangle, \langle v \rangle>$
| `nil`
| `cons` $\langle v \rangle\ \langle v \rangle$

Contexts

$\langle E \rangle$ ::= $\square$
| `for` $\langle E \rangle$ `then` $\langle e \rangle$ `else` $\langle e \rangle$
| `succ` $\langle E \rangle$
| `pred` $\langle E \rangle$
| `isZero` $\langle E \rangle$
| $\langle E \rangle\ \langle e \rangle$
| $\langle v \rangle\ \langle E \rangle$
| $<\langle E \rangle, \langle e \rangle>$
| $<\langle v \rangle, \langle E \rangle>$
| `fst` $\langle E \rangle$
| `snd` $\langle E \rangle$
| `cons` $\langle E \rangle\ \langle e \rangle$
| `cons` $\langle v \rangle\ \langle E \rangle$
| `head` $\langle E \rangle$
| `tail` $\langle E \rangle$
| `isNil` $\langle E \rangle$

2 Typing Rules and Run-Time Semantics

(T-TRUE)

$$\Gamma \vdash \text{true} : \texttt{Bool}$$

(T-FALSE)

$$\Gamma \vdash \text{false} : \texttt{Bool}$$

(T-IF)

$$\frac{\Gamma \vdash \langle e_1 \rangle : \text{Bool} \quad \langle e_2 \rangle : \langle T \rangle \quad \langle e_3 \rangle : \langle T \rangle}{\Gamma \vdash \text{if } \langle e_1 \rangle \text{ then } \langle e_2 \rangle \text{ else } \langle e_3 \rangle : \langle T \rangle}$$

(T-ZERO)

$$\Gamma \vdash 0 : \texttt{Int}$$

(T-SUCC)

$$\frac{\Gamma \vdash \langle e \rangle : \texttt{Int}}{\Gamma \vdash \texttt{succ}\ \langle e \rangle : \texttt{Int}}$$

(T-PRED)

$$\frac{\Gamma \vdash \langle e \rangle : \texttt{Int}}{\Gamma \vdash \texttt{pred}\ \langle e \rangle : \texttt{Int}}$$

(T-ISZERO)

$$\frac{\Gamma \vdash \langle e \rangle : \texttt{Int}}{\Gamma \vdash \texttt{isZero}\ \langle e \rangle : \texttt{Bool}}$$

(T-VAR)

$$\Gamma, x : \langle T \rangle \vdash x : \langle T \rangle$$

(T-LAMBDA)

$$\frac{\Gamma, x : \langle T_1 \rangle \vdash e : \langle T_2 \rangle}{\Gamma \vdash \lambda \langle x \rangle . \langle e \rangle : \langle T_1 \rangle \to \langle T_2 \rangle}$$

(T-APP)

$$\frac{\Gamma \vdash e_1 : \langle T_1 \rangle \to \langle T_2 \rangle \quad \Gamma \vdash e_2 : \langle T_1 \rangle}{\Gamma \vdash \langle e_1 \rangle\ \langle e_2 \rangle : \langle T_2 \rangle}$$

(T-NIL)

$$\Gamma \vdash \texttt{nil} : \texttt{List}\ \langle T \rangle$$

(T-CONS)

$$\frac{\Gamma \vdash \langle e_1 \rangle : \langle T \rangle \quad \Gamma \vdash \langle e_2 \rangle : \texttt{List}\ \langle T \rangle}{\Gamma \vdash \texttt{cons}\ \langle e_1 \rangle\ \langle e_2 \rangle : \texttt{List}\ \langle T \rangle}$$

(T-HEAD)

$$\frac{\Gamma \vdash \langle e \rangle : \texttt{List}\ \langle T \rangle}{\Gamma \vdash \texttt{head}\ \langle e \rangle : \langle T \rangle}$$

(T-TAIL)

$$\frac{\Gamma \vdash \langle e \rangle : \texttt{List}\ \langle T \rangle}{\Gamma \vdash \texttt{tail}\ \langle e \rangle : \texttt{List}\ \langle T \rangle}$$

(T-FST)

$$\frac{\Gamma \vdash \langle e \rangle : \langle T_1 \rangle \times \langle T_2 \rangle}{\Gamma \vdash \texttt{fst}\ \langle e \rangle : \langle T_1 \rangle}$$

(T-SND)

$$\frac{\Gamma \vdash \langle e \rangle : \langle T_1 \rangle \times \langle T_2 \rangle}{\Gamma \vdash \texttt{snd}\ \langle e \rangle : \langle T_2 \rangle}$$

(T-ISNIL)

$$\frac{\Gamma \vdash \langle e \rangle : \texttt{List}\ \langle T \rangle}{\Gamma \vdash \texttt{isNil}\ \langle e \rangle : \texttt{Bool}}$$

(T-ERROR)

$$\Gamma \vdash \langle er \rangle : \langle T \rangle$$

(R-IF-TRUE)
$$\text{if } \texttt{true} \text{ then } \langle e_1 \rangle \text{ else } \langle e_2 \rangle \to \langle e_1 \rangle$$

(R-IF-FALSE)
$$\text{if } \texttt{false} \text{ then } \langle e_1 \rangle \text{ else } \langle e_2 \rangle \to \langle e_2 \rangle$$

(R-PRED-ZERO)
$$\texttt{pred } 0 \to \langle er \rangle$$

(R-PRED-SUCC)
$$\texttt{pred (succ } \langle e \rangle\texttt{)} \to \langle e \rangle$$

(R-ISZERO-ZERO)
$$\texttt{isZero } 0 \to \texttt{true}$$

(R-ISZERO-SUCC)
$$\texttt{isZero (succ } \langle e \rangle\texttt{)} \to \texttt{false}$$

(R-APP)
$$(\lambda \langle x \rangle . \langle e \rangle)\ \langle v \rangle \to \langle e \rangle \left[\langle v \rangle \middle/ \langle x \rangle \right]$$

(R-HEAD-NIL)
$$\texttt{head nil} \to \langle er \rangle$$

(R-HEAD-CONS)
$$\texttt{head (cons } \langle v_1 \rangle\ \langle v_2 \rangle\texttt{)} \to \langle v_1 \rangle$$

(R-FST)
$$\texttt{fst } < \langle v_1 \rangle, \langle v_2 \rangle > \to \langle v_1 \rangle$$

(R-SND)
$$\texttt{snd } < \langle v_1 \rangle, \langle v_2 \rangle > \to \langle v_2 \rangle$$

(R-TAIL-NIL)
$$\texttt{tail nil} \to \langle er \rangle$$

(R-TAIL-CONS)
$$\texttt{tail (cons } \langle v_1 \rangle\ \langle v_2 \rangle\texttt{)} \to \langle v_2 \rangle$$

(R-ISNIL-NIL)
$$\texttt{isNil nil} \to \texttt{true}$$

(R-ISNIL-CONS)
$$\texttt{isNil (cons } \langle v_1 \rangle\ \langle v_2 \rangle\texttt{)} \to false$$

(R-CTX)
$$\frac{\langle e \rangle \to \langle e' \rangle}{\texttt{E}\left[\langle e \rangle\right] \to \texttt{E}\left[\langle e' \rangle\right]}$$

(R-ERROR)
$$\frac{}{\texttt{E}\left[\langle er \rangle\right] \to \langle er \rangle}$$

3 Proving Type Soundness

Type soundness is, generally speaking, the idea that the actual types within a program are the same as the computed types from the type system. Proving the type soundness of our language can be done first proving the progress theorem and the type preservation theorem.

3.1 Progress Theorem

Before we begin proving progress on our language, we first want to be able to reason about the possible shapes of the canonical forms of our types.

3.1.1 Canonical Form Lemmas

Lemma 3.0.1. (Int)

$$\Gamma \vdash \langle e \rangle : \text{Int} \land \langle e \rangle = \langle v \rangle \Rightarrow \langle e \rangle = 0 \lor \langle e \rangle = \text{succ } \langle v \rangle$$

Proof. Let us perform a case analysis on $\Gamma \vdash \langle e \rangle : \text{Int}$ and show it to be true by looking at our aforementioned typing rules.

Cases:

1. $\langle e \rangle = 0$
   a. Trivially true our language definition of values.
2. $\langle e \rangle = \text{succ } \langle e \rangle$
   a. We apply $\langle e \rangle = \langle v \rangle$
   b. Now we have $\text{succ } \langle e \rangle \Rightarrow \text{succ } \langle v \rangle$
   c. Trivially true our language definition of values.

□

Lemma 3.0.2. (Bool)

$$\Gamma \vdash \langle e \rangle : \text{Bool} \land \langle e \rangle = \langle v \rangle \Rightarrow \langle e \rangle = \text{true} \lor \langle e \rangle = \text{false}$$

Proof. Let us perform a case analysis on $\Gamma \vdash \langle e \rangle : \text{Bool}$ and show it to be true by looking at our aforementioned typing rules.

Cases:

1. $\langle e \rangle = \text{true}$
    a. Trivially true our language definition of values.
2. $\langle e \rangle = \text{false}$
    a. Trivially true our language definition of values.

□

Lemma 3.0.3. (List)

$$\Gamma \vdash \langle e \rangle : \text{List } \langle T \rangle \wedge \langle e \rangle = \langle v \rangle \Rightarrow \langle e \rangle = \text{nil} \vee \langle e \rangle = \text{cons } \langle v_1 \rangle \; \langle v_2 \rangle$$

Proof. Let us perform a case analysis on $\Gamma \vdash \langle e \rangle : \text{List } \langle T \rangle$ and show it to be true by looking at our aforementioned typing rules.

Cases:

1. $\langle e \rangle = \text{nil}$
    a. Trivially true our language definition of values.
2. $\langle e \rangle = \text{cons } \langle e_1 \rangle \; \langle e_2 \rangle$
    a. We apply $\langle e \rangle = \langle v \rangle$
    b. Now we have cons $\langle e_1 \rangle \; \langle e_2 \rangle \Rightarrow$ cons $\langle v_1 \rangle \; \langle v_2 \rangle$
    c. Trivially true our language definition of values.

□

Lemma 3.0.4. (Pair)

$$\Gamma \vdash \langle e \rangle : \langle T_1 \rangle \times \langle T_2 \rangle \wedge \langle e \rangle = \langle v \rangle \Rightarrow \langle e \rangle = <\langle v_1 \rangle, \langle v_2 \rangle>$$

Proof. Let us perform a case analysis on $\Gamma \vdash \langle e \rangle : \langle T_1 \rangle \times \langle T_2 \rangle$ and show it to be true by looking at our aforementioned typing rules.

Cases:

1. $<\langle e_1 \rangle, \langle e_2 \rangle>$
    a. We apply $\langle e \rangle = \langle v \rangle$
    b. Now we have $\langle e \rangle = <\langle v_1 \rangle, \langle v_2 \rangle>$
    c. Trivially true our language definition of values.

□

Lemma 3.0.5. (Lambda)

$$\Gamma \vdash \langle e \rangle : \langle T_1 \rangle \to \langle T_2 \rangle \wedge \langle e \rangle = \langle v \rangle \Rightarrow \langle e \rangle = \lambda \langle x \rangle . \langle e \rangle$$

Proof. Let us perform a case analysis on $\Gamma \vdash \langle e \rangle : \langle T_1 \rangle \to \langle T_2 \rangle$ and show it to be true by looking at our aforementioned typing rules.

Cases:

1. $\langle e \rangle = \lambda \langle x \rangle . \langle e \rangle$
   a. Trivially true our language definition of values.

□

3.1.2 Proving Progress Theorem

We can now use these canonical form lemmas to help us further prove progress of our language. Our strategy will be to prove progress individually for each expression in our language.

First, let us define what it means for an expression $\langle e \rangle$ to progress:

$$\langle e \rangle \text{ progress} \equiv \langle e \rangle = \langle v \rangle \vee \langle e \rangle \to \langle e' \rangle$$

Using this definition, we can simplify what we mean when we try to say that an expression $\langle e \rangle$ progresses.

Let's start by showing progress on the trivial expressions.

Theorem 3.1. (progress-true)

$$\langle e \rangle \text{ progress } \wedge \langle e \rangle = \text{true}$$

Proof. Since $\langle e \rangle$ is a value, we know that true progresses and this is trivially true. □

Theorem 3.2. (progress-false)

$$\langle e \rangle \text{ progress } \wedge \langle e \rangle = \text{false}$$

Proof. Since $\langle e \rangle$ is a value, we know that false progresses and this is trivially true. □

Theorem 3.3. (progress-zero)

$$\langle e \rangle \text{ progress } \wedge \langle e \rangle = 0$$

Proof. Since $\langle e \rangle$ is a value, we know that 0 progresses and this is trivially true. $\square$

Theorem 3.4. (progress-nil)

$$\langle e \rangle \text{ progress } \wedge \langle e \rangle = \text{nil}$$

Proof. Since $\langle e \rangle$ is a value, we know that nil progresses and this is trivially true. $\square$

Theorem 3.5. (progress-error)

$$\langle e \rangle \text{ progress } \wedge \langle e \rangle = \langle er \rangle$$

Proof. Since $\langle er \rangle$ is an error, we know that $\langle er \rangle$ progresses and this is trivially true. $\square$

Next, we move on to showing progress on some more involved expressions.

Theorem 3.6. (progress-succ)

$$\langle e \rangle \text{ progress } \Rightarrow \text{succ } \langle e \rangle \text{ progress}$$

Proof. Let us perform a case analysis on $\langle e \rangle$ progress in the context of the progress theorem to show that this form of expression progresses in our language.

Cases:

1. $\langle e \rangle = \langle v \rangle$
   a. $\text{succ } \langle e \rangle \Rightarrow \text{succ } \langle v \rangle = \langle v \rangle$ $\blacksquare$
2. $\langle e \rangle = \langle er \rangle$
   a. $\text{succ } \langle e \rangle \Rightarrow \text{succ } \langle er \rangle \xrightarrow{\text{R-ERROR}} \langle er \rangle$ $\blacksquare$
3. $\langle e \rangle \rightarrow \langle e' \rangle$
   a. $\text{succ } \langle e \rangle \xrightarrow{\text{R-CTX}} \text{succ } \langle e' \rangle$ $\blacksquare$

$\square$

Theorem 3.7. (progress-pred)

$$\langle e \rangle \text{ progress } \Rightarrow \text{pred } \langle e \rangle \text{ progress}$$

Proof. Let us perform a case analysis on $\langle e \rangle$ progress in the context of the progress theorem to show that this form of expression progresses in our language.

Cases:

1. $\langle e \rangle = \langle v \rangle$
   a. $\text{pred } \langle e \rangle \Rightarrow \text{pred } \langle v \rangle \xrightarrow{\text{T-PRED}} \langle v \rangle : \text{Int}$
   b. Using the canonical form lemma for Int (3.0.1) we know $\langle v \rangle$ can hold the possible shapes of:
      i. $\langle v \rangle = 0 \Rightarrow \text{pred } 0 \xrightarrow{\text{R-PRED-ZERO}} \langle er \rangle$ ■
      ii. $\langle v \rangle = \text{succ } \langle v \rangle \Rightarrow \text{pred } (\text{succ } \langle v \rangle) \xrightarrow{\text{R-PRED-PRED}} \langle v \rangle$ ■
2. $\langle e \rangle = \langle er \rangle$
   a. $\text{pred } \langle e \rangle \Rightarrow \text{pred } \langle er \rangle \xrightarrow{\text{R-ERROR}} \langle er \rangle$ ■
3. $\langle e \rangle \to \langle e' \rangle$
   a. $\text{pred } \langle e \rangle \xrightarrow{\text{R-CTX}} \text{pred } \langle e' \rangle$ ■

□

Theorem 3.8. (progress-isZero)

$$\langle e \rangle \text{ progress } \Rightarrow \text{isZero } \langle e \rangle \text{ progress}$$

Proof. Let us perform a case analysis on $\langle e \rangle$ progress in the context of the progress theorem to show that this form of expression progresses in our language.

Cases:

1. $\langle e \rangle = \langle v \rangle$
   a. $\text{isZero } \langle e \rangle \Rightarrow \text{isZero } \langle v \rangle \xrightarrow{\text{T-ISZERO}} \langle v \rangle : \text{Int}$
   b. Using the canonical form lemma for Int (3.0.1) we know $\langle v \rangle$ can hold the possible shapes of:
      i. $\langle v \rangle = 0 \Rightarrow \text{isZero } 0 \xrightarrow{\text{R-ISZERO-ZERO}} \text{true}$ ■
      ii. $\langle v \rangle = \text{succ } \langle v \rangle \Rightarrow \text{isZero } (\text{succ } \langle v \rangle) \xrightarrow{\text{R-ISZERO-SUCC}}$ false ■
2. $\langle e \rangle = \langle er \rangle$

   a. isZero $\langle e \rangle \Rightarrow$ isZero $\langle er \rangle \xrightarrow{\text{R-ERROR}} \langle er \rangle$ ■

3. $\langle e \rangle \rightarrow \langle e' \rangle$

   a. isZero $\langle e \rangle \xrightarrow{\text{R-CTX}}$ isZero $\langle e' \rangle$ ■

□

Theorem 3.9. (progress-isNil)

$$\langle e \rangle \text{ progress } \Rightarrow \text{isNil } \langle e \rangle \text{ progress}$$

Proof. Let us perform a case analysis on $\langle e \rangle$ progress in the context of the progress theorem to show that this form of expression progresses in our language.

Cases:

1. $\langle e \rangle = \langle v \rangle$

   a. isNil $\langle e \rangle \Rightarrow$ isNil $\langle v \rangle \xrightarrow{\text{T-ISNIL}} \langle v \rangle : \text{List}\langle T \rangle$

   b. Using the canonical form lemma for List $\langle T \rangle$ (3.0.3) we know $\langle v \rangle$ can hold the possible shapes of:

      i. $\langle v \rangle =$ nil $\Rightarrow$ isNil nil $\xrightarrow{\text{R-ISNIL-NIL}}$ true ■

      ii. $\langle v \rangle =$ cons $\langle v_1 \rangle \langle v_2 \rangle \Rightarrow$
      isNil (cons $\langle v_1 \rangle \langle v_2 \rangle) \xrightarrow{\text{R-ISNIL-CONS}}$ false ■

2. $\langle e \rangle = \langle er \rangle$

   a. isNil $\langle e \rangle \Rightarrow$ isNil $\langle er \rangle \xrightarrow{\text{R-ERROR}} \langle er \rangle$ ■

3. $\langle e \rangle \rightarrow \langle e' \rangle$

   a. isNil $\langle e \rangle \xrightarrow{\text{R-CTX}}$ isNil $\langle e' \rangle$ ■

□

Theorem 3.10. (progress-fst)

$$\langle e \rangle \text{ progress } \Rightarrow \text{fst } \langle e \rangle \text{ progress}$$

Proof. Let us perform a case analysis on $\langle e \rangle$ progress in the context of the progress theorem to show that this form of expression progresses in our language.

Cases:

1. $\langle e \rangle = \langle v \rangle$

   a. fst $\langle e\rangle \Rightarrow$ fst $\langle v\rangle \xrightarrow{\text{T-FST}} \langle v\rangle : <\langle T_1\rangle \times \langle T_2\rangle>$

   b. Using the canonical form lemma for $<\langle T_1\rangle \times \langle T_2\rangle>$ (3.0.4) we know $\langle v\rangle$ can hold the possible shapes of:

      i. $\langle v\rangle = <\langle v_1\rangle, \langle v_2\rangle> \Rightarrow$ fst $<\langle v_1\rangle, \langle v_2\rangle> \xrightarrow{\text{R-FST}} \langle v_1\rangle$ ■

2. $\langle e\rangle = \langle er\rangle$

   a. fst $\langle e\rangle \Rightarrow$ fst $\langle er\rangle \xrightarrow{\text{R-ERROR}} \langle er\rangle$ ■

3. $\langle e\rangle \to \langle e'\rangle$

   a. fst $\langle e\rangle \xrightarrow{\text{R-CTX}}$ fst $\langle e'\rangle$ ■

□

Theorem 3.11. (progress-snd)

$$\langle e\rangle \text{ progress } \Rightarrow \text{snd } \langle e\rangle \text{ progress}$$

Proof. Let us perform a case analysis on $\langle e\rangle$ progress in the context of the progress theorem to show that this form of expression progresses in our language.

Cases:

1. $\langle e\rangle = \langle v\rangle$

   a. snd $\langle e\rangle \Rightarrow$ snd $\langle v\rangle \xrightarrow{\text{T-SND}} \langle v\rangle : <\langle T_1\rangle \times \langle T_2\rangle>$

   b. Using the canonical form lemma for $<\langle T_1\rangle \times \langle T_2\rangle>$ (3.0.4) we know $\langle v\rangle$ can hold the possible shapes of:

      i. $\langle v\rangle = <\langle v_1\rangle, \langle v_2\rangle> \Rightarrow$ snd $<\langle v_1\rangle, \langle v_2\rangle> \xrightarrow{\text{R-SND}} \langle v_2\rangle$ ■

2. $\langle e\rangle = \langle er\rangle$

   a. snd $\langle e\rangle \Rightarrow$ snd $\langle er\rangle \xrightarrow{\text{R-ERROR}} \langle er\rangle$ ■

3. $\langle e\rangle \to \langle e'\rangle$

   a. snd $\langle e\rangle \xrightarrow{\text{R-CTX}}$ snd $\langle e'\rangle$ ■

□

Theorem 3.12. (progress-head)

$$\langle e\rangle \text{ progress } \Rightarrow \text{head } \langle e\rangle \text{ progress}$$

Proof. Let us perform a case analysis on $\langle e \rangle$ progress in the context of the progress theorem to show that this form of expression progresses in our language.

Cases:

1. $\langle e \rangle = \langle v \rangle$
   a. head $\langle e \rangle \Rightarrow$ head $\langle v \rangle \xrightarrow{\text{T-HEAD}} \langle v \rangle$: List $\langle T \rangle$
   b. Using the canonical form lemma for List $\langle T \rangle$ (3.0.3) we know $\langle v \rangle$ can hold the possible shapes of:
      i. $\langle v \rangle =$ nil $\Rightarrow$ head nil $\xrightarrow{\text{R-HEAD-NIL}} \langle er \rangle$ ■
      ii. $\langle v \rangle =$ cons $\langle v_1 \rangle$ $\langle v_2 \rangle \Rightarrow$
      head (cons $\langle v_1 \rangle$ $\langle v_2 \rangle$) $\xrightarrow{\text{R-HEAD-CONS}} \langle v_1 \rangle$ ■
2. $\langle e \rangle = \langle er \rangle$
   a. head $\langle e \rangle \Rightarrow$ head $\langle er \rangle \xrightarrow{\text{R-ERROR}} \langle er \rangle$ ■
3. $\langle e \rangle \rightarrow \langle e' \rangle$
   a. head $\langle e \rangle \xrightarrow{\text{R-CTX}}$ head $\langle e' \rangle$ ■

□

Theorem 3.13. (progress-tail)

$$\langle e \rangle \text{ progress } \Rightarrow \text{tail } \langle e \rangle \text{ progress}$$

Proof. Let us perform a case analysis on $\langle e \rangle$ progress in the context of the progress theorem to show that this form of expression progresses in our language.

Cases:

1. $\langle e \rangle = \langle v \rangle$
   a. tail $\langle e \rangle \Rightarrow$ tail $\langle v \rangle \xrightarrow{\text{T-TAIL}} \langle v \rangle$: List $\langle T \rangle$
   b. Using the canonical form lemma for List $\langle T \rangle$ (3.0.3) we know $\langle v \rangle$ can hold the possible shapes of:
      i. $\langle v \rangle =$ nil $\Rightarrow$ tail nil $\xrightarrow{\text{R-TAIL-NIL}} \langle er \rangle$ ■
      ii. $\langle v \rangle =$ cons $\langle v_1 \rangle$ $\langle v_2 \rangle \Rightarrow$
      tail (cons $\langle v_1 \rangle$ $\langle v_2 \rangle$) $\xrightarrow{\text{R-TAIL-CONS}} \langle v_2 \rangle$ ■
2. $\langle e \rangle = \langle er \rangle$

a. tail $\langle e \rangle \Rightarrow$ tail $\langle er \rangle \xrightarrow{\text{R-ERROR}} \langle er \rangle$ ■

3. $\langle e \rangle \rightarrow \langle e' \rangle$

a. tail $\langle e \rangle \xrightarrow{\text{R-CTX}}$ tail $\langle e' \rangle$ ■

□

Theorem 3.14. (progress-cons)

$$\langle e_1 \rangle \text{ progress } \wedge \langle e_2 \rangle \text{ progress } \Rightarrow \text{cons } \langle e_1 \rangle \langle e_2 \rangle \text{ progress}$$

Proof. Let us perform a case analysis on $\langle e_1 \rangle$ progress in the context of the progress theorem to show that this form of expression progresses in our language.

Cases:

1. $\langle e_1 \rangle = \langle v \rangle$
   a. cons $\langle e_1 \rangle \langle e_2 \rangle \Rightarrow$ cons $\langle v_1 \rangle \langle e_2 \rangle$
   b. Let us now perform a case analysis on $\langle e_2 \rangle$ progress in this context to show that it progresses alongside $\langle e_1 \rangle$ progress.
      i. $\langle e_2 \rangle = \langle v \rangle$
         a. cons $\langle v_1 \rangle \langle e_2 \rangle \Rightarrow$ cons $\langle v_1 \rangle \langle v_2 \rangle$ ■
      ii. $\langle e_2 \rangle = \langle er \rangle$
         a. cons $\langle v_1 \rangle \langle e_2 \rangle \Rightarrow$ cons $\langle v_1 \rangle \langle er \rangle \xrightarrow{\text{R-ERROR}} \langle er \rangle$ ■
      iii. $\langle e_2 \rangle \rightarrow \langle e_2' \rangle$
         a. cons $\langle v_1 \rangle \langle e_2 \rangle \xrightarrow{\text{R-CTX}}$ cons $\langle v_1 \rangle \langle e_2' \rangle$ ■
2. $\langle e_1 \rangle = \langle er \rangle$
   a. cons $\langle e_1 \rangle \langle e_2 \rangle \Rightarrow$ cons $\langle er \rangle \langle e_2 \rangle \xrightarrow{\text{R-ERROR}} \langle er \rangle$ ■
3. $\langle e_1 \rangle \rightarrow \langle e_1' \rangle$
   a. cons $\langle e_1 \rangle \langle e_2 \rangle \xrightarrow{\text{R-CTX}}$ cons $\langle e_1' \rangle \langle e_2 \rangle$ ■

□

Theorem 3.15. (progress-if)

$$\langle e \rangle \text{ progress } \Rightarrow \text{if } \langle e_1 \rangle \text{ then } \langle e_2 \rangle \text{ else } \langle e_3 \rangle \text{ progress}$$

Proof. Let us perform a case analysis on $\langle e \rangle$ progress in the context of the progress theorem to show that this form of expression progresses in our language.

Cases:

1. $\langle e \rangle = \langle v \rangle$

   a. if $\langle e_1 \rangle$ then $\langle e_2 \rangle$ else $\langle e_3 \rangle \Rightarrow$ if $\langle v \rangle$ then $\langle e_2 \rangle$ else $\langle e_3 \rangle \xrightarrow{\text{T-IF}} \langle v \rangle : \text{Bool}$

   b. Using the canonical form lemma for Bool (3.0.2) we know $\langle v \rangle$ can hold the possible shapes of:

      i. $\langle v \rangle = \text{true} \Rightarrow$ if true then $\langle e_2 \rangle$ else $\langle e_3 \rangle \xrightarrow{\text{R-IF-TRUE}} \langle e_2 \rangle$ ■

      ii. $\langle v \rangle = \text{false} \Rightarrow$ if false then $\langle e_2 \rangle$ else $\langle e_3 \rangle \xrightarrow{\text{R-IF-FALSE}} \langle e_3 \rangle$ ■

2. $\langle e \rangle = \langle er \rangle$

   a. if $\langle e_1 \rangle$ then $\langle e_2 \rangle$ else $\langle e_3 \rangle \Rightarrow$
      if $\langle er \rangle$ then $\langle e_2 \rangle$ else $\langle e_3 \rangle \xrightarrow{\text{R-ERROR}} \langle er \rangle$ ■

3. $\langle e \rangle \rightarrow \langle e' \rangle$

   a. if $\langle e_1 \rangle$ then $\langle e_2 \rangle$ else $\langle e_3 \rangle \xrightarrow{\text{R-CTX}}$ if $\langle e_1' \rangle$ then $\langle e_2 \rangle$ else $\langle e_3 \rangle$ ■

□

Theorem 3.16. (progress-pair)

$$\langle e_1 \rangle \text{ progress } \wedge \langle e_2 \rangle \text{ progress } \Rightarrow <\langle e_1 \rangle, \langle e_2 \rangle> \text{ progress}$$

Proof. Let us perform a case analysis on $\langle e_1 \rangle$ progress in the context of the progress theorem to show that this form of expression progresses in our language.

Cases:

1. $\langle e_1 \rangle = \langle v \rangle$

   a. $<\langle e_1 \rangle, \langle e_2 \rangle> \Rightarrow <\langle v_1 \rangle, \langle e_2 \rangle>$

   b. Let us now perform a case analysis on $\langle e_2 \rangle$ progress in this context to show that it progresses alongside $\langle e_1 \rangle$ progress.

      i. $\langle e_2 \rangle = \langle v \rangle$

a. $<\langle v_1\rangle, \langle e_2\rangle> \Rightarrow <\langle v_1\rangle, \langle v_2\rangle>$ ■

ii. $\langle e_2\rangle = \langle er\rangle$

a. $<\langle v_1\rangle, \langle e_2\rangle> \Rightarrow <\langle v_1\rangle, \langle er\rangle> \xrightarrow{\text{R-ERROR}} \langle er\rangle$ ■

iii. $\langle e_2\rangle \rightarrow \langle e_2'\rangle$

a. $<\langle v_1\rangle, \langle e_2\rangle> \xrightarrow{\text{R-CTX}} <\langle v_1\rangle, \langle e_2'\rangle>$ ■

2. $\langle e_1\rangle = \langle er\rangle$

   a. $<\langle e_1\rangle, \langle e_2\rangle> \Rightarrow <\langle er\rangle, \langle e_2\rangle> \xrightarrow{\text{R-ERROR}} \langle er\rangle$ ■

3. $\langle e_1\rangle \rightarrow \langle e_1'\rangle$

   a. $<\langle e_1\rangle, \langle e_2\rangle> \xrightarrow{\text{R-CTX}} <\langle e_1'\rangle, \langle e_2\rangle>$ ■

□

Theorem 3.17. (progress-app)

$$\langle e_1\rangle \text{ progress } \wedge \langle e_2\rangle \text{ progress } \Rightarrow \langle e_1\rangle\ \langle e_2\rangle \text{ progress}$$

Proof. Let us perform a case analysis on $\langle e_1\rangle$ progress in the context of the progress theorem to show that this form of expression progresses in our language.

Cases:

1. $\langle e_1\rangle = \langle v\rangle$

   a. $\langle e_1\rangle\ \langle e_2\rangle \Rightarrow \langle v_1\rangle\ \langle e_2\rangle$

   b. Let us now perform a case analysis on $\langle e_2\rangle$ progress in this context to show that it progresses alongside $\langle e_1\rangle$ progress.

      i. $\langle e_2\rangle = \langle v\rangle$

         a. $\langle v_1\rangle\ \langle e_2\rangle \Rightarrow \langle v_1\rangle\ \langle v_2\rangle$ ■

         b. From our typing rule (T-APP), we know that $\langle v_1\rangle : \langle T_1\rangle \rightarrow \langle T_2\rangle$ and that $\langle v_2\rangle : \langle T_2\rangle$.

         c. Using the canonical form lemma for $\langle T_1\rangle \rightarrow \langle T_2\rangle$ (3.0.5) we know $\langle v\rangle$ can hold the possible shape of $\langle v\rangle = \lambda\langle x\rangle.\langle e\rangle$

         d. Therefore $\langle v_1\rangle\ \langle v_2\rangle \Rightarrow \lambda\langle x\rangle.\langle e\rangle\ \langle v_2\rangle \xrightarrow{\text{R-APP}} \langle e\rangle\left[\langle v\rangle/\langle x\rangle\right]$ ■

      ii. $\langle e_2\rangle = \langle er\rangle$

a. $\langle v_1 \rangle \; \langle e_2 \rangle \Rightarrow \langle v_1 \rangle \; \langle er \rangle \xrightarrow{\text{R-ERROR}} \langle er \rangle$ ■

iii. $\langle e_2 \rangle \rightarrow \langle e_2' \rangle$

a. $\langle v_1 \rangle \; \langle e_2 \rangle \xrightarrow{\text{R-CTX}} \langle v_1 \rangle \; \langle e_2' \rangle$ ■

2. $\langle e_1 \rangle = \langle er \rangle$

   a. $\langle e_1 \rangle \; \langle e_2 \rangle \Rightarrow \langle er \rangle \; \langle e_2 \rangle \xrightarrow{\text{R-ERROR}} \langle er \rangle$ ■

3. $\langle e_1 \rangle \rightarrow \langle e_1' \rangle$

   a. $\langle e_1 \rangle \; \langle e_2 \rangle \xrightarrow{\text{R-CTX}} \langle e_1' \rangle \; \langle e_2 \rangle$ ■

□

Now that we've shown that our expressions can progress, we must continue to show that given any expression $\langle e \rangle$ that is typeable by some type $\langle T \rangle$, then $\langle e \rangle$ progresses.

Theorem 3.18. (progress-typeable)

$$\Gamma \vdash \langle e \rangle : \langle T \rangle, \text{ then } \langle e \rangle \text{ progresses}$$

Proof. To show this, we will perform an induction analysis on $\Gamma \vdash \langle e \rangle : \langle T \rangle$ using our typing rules.

1. $\Gamma \vdash \text{true} : \text{Bool}$
   Trivially true since Bool is a type. ■

2. $\Gamma \vdash \text{false} : \text{Bool}$
   Trivially true since Bool is a type. ■

3. $\Gamma \vdash 0 : \text{Int}$
   Trivially true since Int is a type. ■

4. $\Gamma, x : \langle T \rangle \vdash x : \langle T \rangle$
   Trivially true since x is typeable by the type $\langle T \rangle$ ■

5. $\Gamma \vdash \texttt{nil} : \texttt{List} \; \langle T \rangle$
   Trivially true since x is typeable by the type List $\langle T \rangle$ ■

6. $$\frac{\Gamma \vdash \langle e_1 \rangle : \text{Bool} \quad \langle e_2 \rangle : \langle T \rangle \quad \langle e_3 \rangle : \langle T \rangle}{\Gamma \vdash \text{if } \langle e_1 \rangle \text{ then } \langle e_2 \rangle \text{ else } \langle e_3 \rangle : \langle T \rangle}$$
By inductive hypothesis, we know that $\langle e_1 \rangle$, $\langle e_2 \rangle$, and $\langle e_3 \rangle$ are all typeable. Specifically, in this case, $\langle e_1 \rangle : \text{Bool}$, $\langle e_2 \rangle : \langle T \rangle$, and $\langle e_2 \rangle : \langle T \rangle$. ■

7. $$\frac{\Gamma \vdash \langle e \rangle : \texttt{Int}}{\Gamma \vdash \texttt{pred } \langle e \rangle : \texttt{Int}}$$
By inductive hypothesis, we know that $\langle e \rangle$ is typeable. Specifically, in this case, $\langle e \rangle : \text{Int}$. ■

8. $$\frac{\Gamma \vdash \langle e \rangle : \texttt{Int}}{\Gamma \vdash \texttt{isZero } \langle e \rangle : \texttt{Bool}}$$
By inductive hypothesis, we know that $\langle e \rangle$ is typeable. Specifically, in this case, $\langle e \rangle : \text{Bool}$. ■

9. $$\frac{\Gamma, x : \langle T_1 \rangle \vdash e : \langle T_2 \rangle}{\Gamma \vdash \lambda \langle x \rangle . \langle e \rangle : \langle T_1 \rangle \to \langle T_2 \rangle}$$
By inductive hypothesis, we know that both $\langle x \rangle$ and $\langle e \rangle$ are typeable. Specifically, in this case, $\langle x \rangle : \langle T_1 \rangle$ and $\langle e \rangle : \langle T_2 \rangle$. ■

10. $$\frac{\Gamma \vdash e_1 : \langle T_1 \rangle \to \langle T_2 \rangle \quad \Gamma \vdash e_2 : \langle T_2 \rangle}{\Gamma \vdash \langle e_1 \rangle \ \langle e_2 \rangle : \langle T_2 \rangle}$$
By inductive hypothesis, we know that both $\langle e_1 \rangle$ and $\langle e_2 \rangle$ are typeable. Specifically, in this case, $\langle e_1 \rangle : \langle T_1 \rangle \to \langle T_2 \rangle$ and $e_2 : \langle T_2 \rangle$. ■

11. $$\frac{\Gamma \vdash \langle e_1 \rangle : \langle T \rangle \quad \Gamma \vdash \langle e_2 \rangle : \texttt{List } \langle T \rangle}{\Gamma \vdash \texttt{cons } \langle e_1 \rangle \ \langle e_2 \rangle : \texttt{List } \langle T \rangle}$$
By inductive hypothesis, we know that both $\langle e_1 \rangle$ and $\langle e_2 \rangle$ are typeable. Specifically, in this case, $\langle e_1 \rangle : \langle T \rangle$ and $\langle e_2 \rangle : \text{List } \langle T \rangle$. ■

12. $$\frac{\Gamma \vdash \langle e \rangle : \texttt{List } \langle T \rangle}{\Gamma \vdash \texttt{head } \langle e \rangle : \langle T \rangle}$$
By inductive hypothesis, we know that $\langle e \rangle$ is typeable. Specifically, in this case, $\langle e \rangle : \text{List } \langle T \rangle$. ■

13. $\dfrac{\Gamma \vdash \langle e \rangle : \texttt{List}\ \langle T \rangle}{\Gamma \vdash \texttt{tail}\ \langle e \rangle : \texttt{List}\ \langle T \rangle}$
By inductive hypothesis, we know that $\langle e \rangle$ is typeable.
Specifically, in this case, $\langle e \rangle$: List $\langle T \rangle$. ■

14. $\dfrac{\Gamma \vdash \langle e \rangle : \langle T_1 \rangle \times \langle T_2 \rangle}{\Gamma \vdash \texttt{fst}\ \langle e \rangle : \langle T_1 \rangle}$
By inductive hypothesis, we know that $\langle e \rangle$ is typeable.
Specifically, in this case, $\langle e \rangle : \langle T_1 \rangle \times \langle T_2 \rangle$. ■

15. $\dfrac{\Gamma \vdash \langle e \rangle : \langle T_1 \rangle \times \langle T_2 \rangle}{\Gamma \vdash \texttt{snd}\ \langle e \rangle : \langle T_2 \rangle}$
By inductive hypothesis, we know that $\langle e \rangle$ is typeable.
Specifically, in this case, $\langle e \rangle : \langle T_1 \rangle \times \langle T_2 \rangle$. ■

16. $\dfrac{\Gamma \vdash \langle e \rangle : \texttt{List}\ \langle T \rangle}{\Gamma \vdash \texttt{isNil}\ \langle e \rangle : \texttt{Bool}}$
By inductive hypothesis, we know that $\langle e \rangle$ is typeable.
Specifically, in this case, $\langle e \rangle$: List $\langle T \rangle$. ■

□

Since we know that for all expressions $\langle e \rangle$ and types $\langle T \rangle$, $\langle e \rangle$ progresses as shown by theorem 3.19 alongside all of our other expression progress theorems, we can conclude that our language has progress.

3.2 Preservation Theorem

A language to have preservation means that for all expressions $\langle e \rangle$ and types $\langle T \rangle$, if $\langle e \rangle$ were to transition in one step to some other expression $\langle e' \rangle$, then its type is preserved.

Theorem 3.19. (progress-typeable)

$$\forall e, e' \in \textit{expressions} \land \forall T \in \textit{types} \tag{1}$$
$$\textit{if } e : T \textit{ and } e \to e' \textit{ then } e' : T \tag{2}$$

Proof. To show this, we will perform an induction analysis on $\Gamma \vdash \langle e \rangle : \langle T \rangle$ using our typing rules.

1. if `true` then $\langle e_1 \rangle$ else $\langle e_2 \rangle \to \langle e_1 \rangle$
   (1) From T-IF, we know that both $\langle e_1 \rangle$ and $\langle e_2 \rangle$ hold the type $\langle T \rangle$.

   (2) As concluded from T-IF, we know that $\langle e_1 \rangle$ on the right hand side already holds the type $\langle T \rangle$.

   From (1) and (2), we know this expression's types are preserved. ■

2. if `false` then $\langle e_1 \rangle$ else $\langle e_2 \rangle \to \langle e_2 \rangle$
   (1) From T-IF, we know that both $\langle e_1 \rangle$ and $\langle e_2 \rangle$ hold the type $\langle T \rangle$.

   (2) As concluded from T-IF, we know that $\langle e_2 \rangle$ on the right hand side already holds the type $\langle T \rangle$.

   From (1) and (2), we know this expression's types are preserved. ■

3. `pred` $0 \to \langle er \rangle$
   (1) From T-ZERO, we know that 0 holds the type `Int`. From T-PRED, we know that since 0 is of type `Int`, then `pred` holds the type `Int`.

   (2) From T-ERROR, we know that `true` holds the type `Bool`

   From (1) and (2), we know this expression's types are preserved. ■

4. `pred` (`succ` $\langle e \rangle) \to \langle e \rangle$
   (1) From T-PRED, we know that `succ` $\langle e \rangle$ holds the type `Int` and that this entire expression holds the type `Int`. From

T-SUCC, we know that $\langle e \rangle$ holds the type `Int`.

(2) Since we know from the left hand side that $\langle e \rangle$ holds the type `Int`, we know that this $\langle e \rangle$ also holds the type `Int` since it is the same $\langle e \rangle$.

From (1) and (2), we know this expression's types are preserved. ∎

5. `isZero 0` $\rightarrow$ `true`
   (1) From T-ZERO, we know that 0 holds the type `Int`. From T-ISZERO, we know that since 0 is of type `Int`, then `isZero` holds the type `Bool`.

   (2) From T-ERROR, we know that $\langle er \rangle$ holds any type $\langle T \rangle$. This means, in this context, it is compatible with our left hand side type of `Bool`.

   From (1) and (2), we know this expression's types are preserved. ∎

6. `isZero (succ` $\langle e \rangle$`)` $\rightarrow$ `false`
   (1) From T-ISZERO, we know that `succ` $\langle e \rangle$ holds the type `Int` and that the entire expression holds the type `Bool`. From T-SUCC, we know that $\langle e \rangle$ holds the type `Int`.

   (2) Since we know from T-FALSE that false holds the type `Bool`, we know this entire expression holds the type `Bool`.

   From (1) and (2), we know this expression's types are preserved. ∎

7. $(\lambda\langle x \rangle.\langle e \rangle)\ \langle v \rangle \rightarrow \langle e \rangle \left[\langle v \rangle \middle/ \langle x \rangle\right]$
   (1) From T-LAMBDA we know that $\lambda\langle x \rangle.\langle e \rangle$ holds the type $\langle T_1 \rangle \rightarrow \langle T_2 \rangle$. From T-APP we know that since $\lambda\langle x \rangle.\langle e \rangle$ holds the type $\langle T_1 \rangle \rightarrow \langle T_2 \rangle$, then $\langle v \rangle$ holds the type $\langle T_1 \rangle$ and that this entire expression holds the type $\langle T_2 \rangle$.

   (2) Using the truths we've derived from (1), we can apply the Substitution Lemma to show that $\langle e \rangle \left[\langle v \rangle \middle/ \langle x \rangle\right]$ holds the type

$\langle T_2 \rangle$.

Recall that the Substitution Lemma states:

$$\Gamma, x : T_1, \vdash e : T_2 \land \vdash e' : T_1 \Rightarrow \Gamma \vdash e\left[e'/x\right] : T_2 \qquad (3)$$

From (1) and (2), we know this expression's types are preserved. ■

8. `head nil` $\rightarrow \langle er \rangle$
   (1) From T-HEAD, we know that nil holds the type `List` $\langle T \rangle$ and that the entire expression holds the type $\langle T \rangle$. From T-NIL, we know that $\langle e \rangle$ holds the type `List` $\langle T \rangle$.

   (2) From T-ERROR, we know that $\langle er \rangle$ holds any type $\langle T \rangle$. This means, in this context, it is compatible with our left hand side type of $\langle T \rangle$.

   From (1) and (2), we know this expression's types are preserved. ■

9. `head (cons` $\langle v_1 \rangle$ $\langle v_2 \rangle$`)` $\rightarrow \langle v_1 \rangle$
   (1) From T-HEAD, we know that `cons` $\langle v_1 \rangle$ $\langle v_2 \rangle$ holds the type `List` $\langle T \rangle$ and that the entire expression holds the type $\langle T \rangle$. From T-CONS, we know that $\langle v_1 \rangle$ holds the type $\langle T \rangle$.

   (2) Since we know from the left hand side that $\langle v_1 \rangle$ holds the type $\langle T \rangle$, we know that this entire expression holds the type $\langle T \rangle$.

   From (1) and (2), we know this expression's types are preserved. ■

10. `fst` $<\langle v_1 \rangle, \langle v_2 \rangle> \rightarrow \langle v_1 \rangle$
    (1) From T-FST, we know that $<\langle v_1 \rangle, \langle v_2 \rangle>$ holds the type $\langle T_1 \rangle \rightarrow \langle T_2 \rangle$ and the entire expression holds the value $\langle T_1 \rangle$.

    (2) Since we know from the left hand side that $\langle v_1 \rangle$ holds the type $\langle T_1 \rangle$, we know that this entire expression holds the type $\langle T_1 \rangle$.

    From (1) and (2), we know this expression's types are preserved. ■

11. `snd` $<\langle v_1\rangle, \langle v_2\rangle> \to \langle v_2\rangle$
(1) From T-FST, we know that $<\langle v_1\rangle, \langle v_2\rangle>$ holds the type $\langle T_1\rangle \to \langle T_2\rangle$ and the entire expression holds the value $\langle T_2\rangle$.

(2) Since we know from the left hand side that $\langle v_2\rangle$ holds the type $\langle T_1\rangle$, we know that this entire expression holds the type $\langle T_2\rangle$.

From (1) and (2), we know this expression's types are preserved. ■

12. `tail nil` $\to \langle er\rangle$
(1) From T-NIL, we know that nil holds the type `List` $\langle T\rangle$. From T-TAIL, we know that since nil is of type `List` $\langle T\rangle$, then `tail` holds the type `List` $\langle T\rangle$.

(2) From T-ERROR, we know that $\langle er\rangle$ holds any type $\langle T\rangle$. This means, in this context, it is compatible with our left hand side type of `List` $\langle T\rangle$.

From (1) and (2), we know this expression's types are preserved. ■

13. `tail (cons` $\langle v_1\rangle\ \langle v_2\rangle$`)` $\to \langle v_2\rangle$
(1) From T-TAIL, we know that `cons` $\langle v_1\rangle\ \langle v_2\rangle$ holds the type `List` $\langle T\rangle$ and that the entire expression holds the type `List` $\langle T\rangle$. From T-CONS, we know that $\langle v_2\rangle$ holds the type `List` $\langle T\rangle$.

(2) Since we know from the left hand side that $\langle v_2\rangle$ holds the type `List` $\langle T\rangle$, we know that this entire expression holds the type `List` $\langle T\rangle$.

From (1) and (2), we know this expression's types are preserved. ■

14. `isNil nil` $\to$ `true`
(1) From T-ISNIL, we know that nil holds the type `List` $\langle T\rangle$ and that the entire expression holds the type `Bool`. From

T-NIL, we know that $\langle e \rangle$ holds the type `List` $\langle T \rangle$.

(2) Since we know from T-FALSE that false holds the type `Bool`, we know this entire expression holds the type `Bool`.

From (1) and (2), we know this expression's types are preserved. ■

15. `isNil` (`cons` $\langle v_1 \rangle$ $\langle v_2 \rangle$) $\rightarrow$ *false*
    (1) From T-ISNIL, we know that `cons` $\langle v_1 \rangle$ $\langle v_2 \rangle$ holds the type `List` $\langle T \rangle$ and that the entire expression holds the type `Bool`. From T-CONS, we know that $\langle v_1 \rangle$ holds the type $\langle T \rangle$.

    (2) Since we know from T-TRUE that true holds the type `Bool`, we know this entire expression holds the type `Bool`.

    From (1) and (2), we know this expression's types are preserved. ■

□

3.3 Finishing Type Soundness Theorem

The type soundness theorem states that

$$\begin{aligned}
&\forall e, e' \in \text{expressions} \wedge \forall T \in \text{types} && (4)\\
&\textit{if } e : T \textit{ and } e \xrightarrow{*} e' \textit{ then } e' \textit{ is either:} && (5)\\
&\text{- } e' \textit{ is a value such that its type is preserved} && (6)\\
&\text{- } e' \textit{ is an error} && (7)\\
&\text{- } \exists e'' \textit{ such that } e' \rightarrow e'' && (8)
\end{aligned}$$

From having shown type soundness and progress to be true within our language, we can clearly observe that our language has type soundness from this theorem.