Assignment 3

Jana Dunfield

due: Wednesday, 22 March 2022

If you wish, you may work in a group of up to 3 on this assignment. Only one member needs to submit the assignment (but make sure at least one of you does), but **each** group member must **also** submit a brief statement, listing all the group members and describing the approximate division of labour between group members, including an estimate of how many hours you spent. For example, if you are in a group of 2, we need one assignment plus two statements, submitted separately.

If you are working alone, you may write an estimate the time you spent, but it is not required (and no separate statement is required).

ID number(s): 20311734

Name(s) (optional): Zhimin Zhao

Late policy

Assignments may be submitted up to 24 hours late with no penalty.

Scanning

Submitting a legible scan is acceptable, but please ensure that the total file size is less than 30 MB. It is preferable to combine pages into a single PDF file.

Note: The question marked with a **+** is a bonus question, and **it can only add up to 5% to your mark**. You can receive 100% without doing it.

1 Typing

In this assignment, we consider the following expressions e, types T (or S), and typing contexts Γ:

$$
\begin{array}{rcl}
e & ::= & \texttt{()} \\
& & \mid n \mid \texttt{(+ } e\ e\texttt{)} \mid \texttt{(- } e\ e\texttt{)} \\
& & \mid \texttt{True} \mid \texttt{False} \mid \texttt{(Ite } e\ e\ e\texttt{)} \\
& & \mid \texttt{(= } e\ e\texttt{)} \mid \texttt{(< } e\ e\texttt{)} \\
& & \mid x \mid \texttt{(Lam } x\ e\texttt{)} \mid \texttt{(Call } e\ e\texttt{)} \\
& & \mid \texttt{(Pair } e\ e\texttt{)} \mid \texttt{(Proj}_1\ e\texttt{)} \mid \texttt{(Proj}_2\ e\texttt{)} \\
& & \mid \texttt{(Inj}_1\ e\texttt{)} \mid \texttt{(Inj}_2\ e\texttt{)} \\
& & \mid \texttt{(Case } e\ \texttt{(}x\ \texttt{=>}\ e\texttt{)}\ \texttt{(}x\ \texttt{=>}\ e\texttt{))}
\end{array}
$$

$$
\begin{array}{rcll}
S, T & ::= & \texttt{unit} & \text{unit type} \\
& & \mid \texttt{int} & \text{type of integers} \\
& & \mid \texttt{bool} & \text{type of booleans} \\
& & \mid S \to T & \text{type of functions on } S \text{ that produce } T \\
& & \mid S \times T & \text{type of pairs of one } S \text{ and one } T \\
& & \mid S + T & \text{type of a left-injected } S \text{ or right-injected } T \\
\Gamma & ::= & \emptyset & \text{empty context} \\
& & \mid \Gamma, x : S & x \text{ has type } S
\end{array}
$$

Consider the following typing rules given in Figure 1.

$\boxed{\Gamma \vdash e : T}$ Under assumptions Γ, expression e has type T

$$\frac{(x : S) \in \Gamma}{\Gamma \vdash x : S}\ \text{type-assum} \qquad \frac{\Gamma, x : S \vdash e : T}{\Gamma \vdash \texttt{(Lam } x\ e\texttt{)} : (S \to T)}\ \to\text{Intro} \qquad \frac{\Gamma \vdash e_1 : (S \to T) \qquad \Gamma \vdash e_2 : S}{\Gamma \vdash \texttt{(Call } e_1\ e_2\texttt{)} : T}\ \to\text{Elim}$$

$$\frac{}{\Gamma \vdash \texttt{()} : \texttt{unit}}\ \text{unitIntro} \qquad \frac{}{\Gamma \vdash \texttt{True} : \texttt{bool}}\ \text{type-true} \qquad \frac{}{\Gamma \vdash \texttt{False} : \texttt{bool}}\ \text{type-false}$$

$$\frac{}{\Gamma \vdash n : \texttt{int}}\ \text{intIntro} \qquad \frac{\Gamma \vdash e_1 : \texttt{int} \qquad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash \texttt{(+ } e_1\ e_2\texttt{)} : \texttt{int}}\ \text{type-add} \qquad \frac{\Gamma \vdash e_1 : \texttt{int} \qquad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash \texttt{(- } e_1\ e_2\texttt{)} : \texttt{int}}\ \text{type-sub}$$

$$\frac{\Gamma \vdash e_1 : \texttt{int} \qquad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash \texttt{(= } e_1\ e_2\texttt{)} : \texttt{bool}}\ \text{type-equals} \qquad \frac{\Gamma \vdash e_1 : \texttt{int} \qquad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash \texttt{(< } e_1\ e_2\texttt{)} : \texttt{bool}}\ \text{type-lt}$$

$$\frac{\Gamma \vdash e : \texttt{bool} \qquad \Gamma \vdash e_{\text{then}} : T \qquad \Gamma \vdash e_{\text{else}} : T}{\Gamma \vdash \texttt{(Ite } e\ e_{\text{then}}\ e_{\text{else}}\texttt{)} : T}\ \text{type-ite}$$

$$\frac{\Gamma \vdash e_1 : S_1}{\Gamma \vdash \texttt{(Inj}_1\ e_1\texttt{)} : (S_1 + S_2)}\ +\text{Intro1} \qquad \frac{\Gamma \vdash e_2 : S_2}{\Gamma \vdash \texttt{(Inj}_2\ e_2\texttt{)} : (S_1 + S_2)}\ +\text{Intro2}$$

$$\frac{\Gamma \vdash e : (S_1 + S_2) \qquad \Gamma, x_1 : S_1 \vdash e_1 : T \qquad \Gamma, x_2 : S_2 \vdash e_2 : T}{\Gamma \vdash \texttt{(Case } e\ \texttt{(}x_1\ \texttt{=>}\ e_1\texttt{)}\ \texttt{(}x_2\ \texttt{=>}\ e_2\texttt{))} : T}\ +\text{Elim}$$

$$\frac{\Gamma \vdash e_1 : S_1 \qquad \Gamma \vdash e_2 : S_2}{\Gamma \vdash \texttt{(Pair } e_1\ e_2\texttt{)} : (S_1 \times S_2)}\ \times\text{Intro} \qquad \frac{\Gamma \vdash e : (S_1 \times S_2)}{\Gamma \vdash \texttt{(Proj}_1\ e\texttt{)} : S_1}\ \times\text{Elim1} \qquad \frac{\Gamma \vdash e : (S_1 \times S_2)}{\Gamma \vdash \texttt{(Proj}_2\ e\texttt{)} : S_2}\ \times\text{Elim2}$$

Figure 1 Typing with functions ($\to$), integers, booleans, sums ($+$), and pairs ($\times$)

Question 1(a). Let $\Gamma = (x : \mathsf{bool})$. Complete the following typing derivation.

$$\dfrac{\dfrac{(x : \mathsf{bool}) \in \Gamma}{\Gamma \vdash x : \mathsf{bool}}\,\text{type-assum} \quad \dfrac{}{\Gamma \vdash 2 : \mathsf{int}}\,\text{intIntro} \quad \dfrac{}{\Gamma \vdash 8 : \mathsf{int}}\,\text{intIntro}}{\Gamma \vdash (\texttt{Ite}\ x\ 2\ 8) : \mathsf{int}}\,\text{type-ite}$$

Question 1(b). Let $\Gamma = (x : (\mathsf{int} + \mathsf{bool}))$. Complete the following typing derivation.

$$\dfrac{\dfrac{(x : (\mathsf{int} + \mathsf{bool})) \in \Gamma}{\Gamma \vdash x : (\mathsf{int} + \mathsf{bool})}\,\text{type-assum} \quad \dfrac{}{\Gamma, z : \mathsf{int} \vdash \texttt{False} : \mathsf{bool}}\,\text{type-false} \quad \dfrac{(y : \mathsf{bool}) \in \Gamma}{\Gamma, y : \mathsf{bool} \vdash y : \mathsf{bool}}\,\text{type-assum}}{\Gamma \vdash (\texttt{Case}\ x\ (z \texttt{=>}\ \texttt{False})\ (y \texttt{=>}\ y)) : \mathsf{bool}}\,+\text{Elim}$$

Question 1(c). Booleans are not really necessary, because we can write $(\texttt{Inj}_1\ ())$ instead of `True`, $(\texttt{Inj}_2\ ())$ instead of `False`, and `Case` instead of `Ite`. Translate the expression from 1(a), $(\texttt{Ite}\ x\ 2\ 8)$, into an expression that has type `int` under the typing context

$$x : (\mathsf{unit} + \mathsf{unit})$$

and which would step to 2 if x were replaced with $(\texttt{Inj}_1\ ())$, and to 8 if x were replaced with $(\texttt{Inj}_2\ ())$. **Hint:** think about the derivation in 1(b).

$$(\texttt{Case}\ x\ (x_1 \texttt{=>}\ 2)\ (x_2 \texttt{=>}\ 8))$$

2 Mirror World

Consider the sequent calculus rules in Figure 2.

$\boxed{\Gamma \vdash A \text{ true}}$ Under assumptions Γ, formula A is true

$$\frac{x\big[A \text{ true}\big] \in \Gamma}{\Gamma \vdash A \text{ true}}\ \text{sc-assum} \qquad \frac{\Gamma, x\big[A \text{ true}\big] \vdash B \text{ true}}{\Gamma \vdash (A \supset B) \text{ true}}\ \text{sc-}\supset\text{Intro} \qquad \frac{\Gamma \vdash A \supset B \text{ true} \qquad \Gamma \vdash A \text{ true}}{\Gamma \vdash B \text{ true}}\ \text{sc-}\supset\text{Elim}$$

$$\frac{}{\Gamma \vdash \text{True true}}\ \text{sc-TrueIntro}$$

$$\frac{\Gamma \vdash A \text{ true}}{\Gamma \vdash A \vee B \text{ true}}\ \text{sc-}\vee\text{Intro1} \qquad \frac{\Gamma \vdash B \text{ true}}{\Gamma \vdash A \vee B \text{ true}}\ \text{sc-}\vee\text{Intro2}$$

$$\frac{\Gamma \vdash A \vee B \text{ true} \qquad \Gamma, x\big[A \text{ true}\big] \vdash C \text{ true} \qquad \Gamma, y\big[B \text{ true}\big] \vdash C \text{ true}}{\Gamma \vdash C \text{ true}}\ \text{sc-}\vee\text{Elim}$$

$$\frac{\Gamma \vdash A \text{ true} \qquad \Gamma \vdash B \text{ true}}{\Gamma \vdash A \wedge B \text{ true}}\ \text{sc-}\wedge\text{Intro} \qquad \frac{\Gamma \vdash A \wedge B \text{ true}}{\Gamma \vdash A \text{ true}}\ \text{sc-}\wedge\text{Elim1} \qquad \frac{\Gamma \vdash A \wedge B \text{ true}}{\Gamma \vdash B \text{ true}}\ \text{sc-}\wedge\text{Elim2}$$

Figure 2 Sequent calculus, with $\supset$, True, $\vee$, and $\wedge$

Some of the typing rules from Figure 1 have a curious property: if we change some of the meta-variables, remove the expression and colon, translate the types, and add the word true, we get a rule in Figure 2.

For example, the rule $\times$Elim1 becomes the rule sc-$\wedge$Elim1:

$$\frac{\Gamma \vdash e : S_1 \times S_2}{\Gamma \vdash \text{proj}_1\ e : S_1} \implies \frac{\Gamma \vdash e : A_1 \times A_2}{\Gamma \vdash \text{proj}_1\ e : A_1} \implies \frac{\Gamma \vdash A_1 \times A_2}{\Gamma \vdash A_1} \implies \frac{\Gamma \vdash A_1 \wedge A_2}{\Gamma \vdash A_1} \implies \frac{\Gamma \vdash A_1 \wedge A_2 \text{ true}}{\Gamma \vdash A_1 \text{ true}}$$

Here we translated the type $S_1 \times S_2$ to $A_1 \wedge A_2$ by replacing $\times$ with $\wedge$. The types can be translated as follows:

unit	becomes	True
$\times$	becomes	$\wedge$
$+$	becomes	$\vee$
$\rightarrow$	becomes	$\supset$

Assumptions need some extra work; for example, in translating $\rightarrow$Intro, $x : S$ becomes $x\big[S \text{ true}\big]$:

$$\frac{\Gamma, x : S \vdash e : T}{\Gamma \vdash (\text{Lam } x\ e) : S \rightarrow T} \implies \frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash (\text{Lam } x\ e) : A \rightarrow B} \implies \frac{\Gamma, x\big[A \text{ true}\big] \vdash B \text{ true}}{\Gamma \vdash (A \supset B) \text{ true}}\ \text{sc-}\supset\text{Intro}$$

Not all the typing rules in Figure 1 have meaningful translations. The rules involving arithmetic, among others, do not lead to anything interesting.

Question 2(a). Pick exactly three rules from Figure 1 such that each has a meaningful translation (following the procedure above) to exactly one corresponding rule in Figure 2. Do **not** pick $\times$Elim1 or $\rightarrow$Intro. In the table below, fill in rows (1), (2) and (3). In the first column of each row, write a rule you chose; and, in the second column, write its corresponding translation.

Figure 1 rule	Figure 2 rule
$\rightarrow$Intro	sc-$\supset$Intro
$\times$Elim1	sc-$\wedge$Elim1
(1) type-assum	sc-assum
(2) type-true	unitIntro
(3) $\rightarrow$Elim	sc-$\supset$Elim

Question 2(b)+ (up to 5% bonus).

A single $\wedge$-elimination: We have two elimination rules for $\wedge$, which separately extract a subformula. Design a *single* sequent-calculus elimination rule for $\wedge$. **Hint:** think about the structure of sc-$\vee$Elim. **Second hint:** think about the connection between $(P_1 \wedge P_2) \supset Q$ and $P_1 \supset (P_2 \supset Q)$, and the shape of a derivation of $\emptyset \vdash P_1 \supset (P_2 \supset Q)$ true: how many assumptions are needed within that derivation?

$$\frac{\Gamma \vdash (A \wedge B)\ \mathsf{true} \qquad \Gamma, x\big[A\ \mathsf{true}\big], y\big[B\ \mathsf{true}\big] \vdash C\ \mathsf{true}}{\Gamma \vdash C\ \mathsf{true}}\ \text{sc-}\wedge\text{Elim}$$

A single $\times$-elimination rule: Translate your new $\wedge$-elimination rule into a typing rule. You will need to extend the grammar of expressions e.

$e ::= \ldots \mid (\mathtt{Split}\ e(< x, x >\Rightarrow e))$ OR $e ::= \ldots \mid (\mathtt{Split}\ e(< x_1, x_2 >\Rightarrow e'))$

$$\frac{\Gamma \vdash e : (S_1 \times S_2) \qquad \Gamma, x_1 : S_1, x_2 : S_2 \vdash e' : T}{\Gamma \vdash \mathtt{Split}\ e(< x_1, x_2 >\Rightarrow e') : T}\ \text{sc-}\times\text{Elim}$$

Small-step semantics: Extend the small-step semantics (see next page) with a reduction rule for your new expression form, and extend the grammar $\mathcal{C}$ of contexts as appropriate.

$C ::= \ldots \mid (\mathtt{Split}\ C(< x, x >\Rightarrow e))$

$$\frac{}{\mathtt{Split}\ (\mathtt{Pair}\ v_1\ v_2)(< x_1, x_2 >\Rightarrow e) \mapsto_R [v_1/x_1, v_2/x_2]e}\ \text{red-split}$$

3 Progress

For this question, we consider the following small-step semantics.

Values $v ::= ()$
$\mid n$
$\mid \texttt{True} \mid \texttt{False}$
$\mid x \mid (\texttt{Lam}\ x\ e)$
$\mid (\texttt{Pair}\ v\ v)$
$\mid (\texttt{Inj}_1\ v) \mid (\texttt{Inj}_2\ v)$

Evaluation contexts $\mathcal{C} ::= [\,]$
$\mid (\texttt{+}\ \mathcal{C}\ e) \mid (\texttt{+}\ v\ \mathcal{C})$
$\mid (\texttt{-}\ \mathcal{C}\ e) \mid (\texttt{-}\ v\ \mathcal{C})$
$\mid (\texttt{Ite}\ \mathcal{C}\ e\ e)$
$\mid (\texttt{=}\ \mathcal{C}\ e) \mid (\texttt{=}\ v\ \mathcal{C})$
$\mid (\texttt{<}\ \mathcal{C}\ e) \mid (\texttt{<}\ v\ \mathcal{C})$
$\mid (\texttt{Call}\ \mathcal{C}\ e)$
$\mid (\texttt{Call}\ v\ \mathcal{C})$
$\mid (\texttt{Pair}\ \mathcal{C}\ e) \mid (\texttt{Pair}\ v\ \mathcal{C})$
$\mid (\texttt{Proj}_1\ \mathcal{C}) \mid (\texttt{Proj}_2\ \mathcal{C})$
$\mid (\texttt{Inj}_1\ \mathcal{C}) \mid (\texttt{Inj}_2\ \mathcal{C})$
$\mid (\texttt{Case}\ \mathcal{C}\ (x\ \texttt{=>}\ e)\ (x\ \texttt{=>}\ e))$

$\boxed{e \mapsto_R e'}$ Expression e reduces to e'

$$\frac{}{(\texttt{+}\ n_1\ n_2) \mapsto_R (n_1 + n_2)}\ \text{red-add} \qquad \frac{}{(\texttt{-}\ n_1\ n_2) \mapsto_R (n_1 - n_2)}\ \text{red-sub}$$

$$\frac{}{(\texttt{=}\ n_1\ n_2) \mapsto_R (n_1 = n_2)}\ \text{red-equals} \qquad \frac{}{(\texttt{<}\ n_1\ n_2) \mapsto_R (n_1 < n_2)}\ \text{red-lessthan}$$

$$\frac{}{(\texttt{Ite True}\ e_{\text{then}}\ e_{\text{else}}) \mapsto_R e_{\text{then}}}\ \text{red-ite-then} \qquad \frac{}{(\texttt{Ite True}\ e_{\text{then}}\ e_{\text{else}}) \mapsto_R e_{\text{then}}}\ \text{red-ite-then}$$

$$\frac{}{(\texttt{Call}\ (\texttt{Lam}\ x\ e)\ v) \mapsto_R [v/x]e}\ \text{red-beta}$$

$$\frac{}{(\texttt{Proj}_1\ (\texttt{Pair}\ v_1\ v_2)) \mapsto_R v_1}\ \text{red-proj1} \qquad \frac{}{(\texttt{Proj}_2\ (\texttt{Pair}\ v_1\ v_2)) \mapsto_R v_2}\ \text{red-proj2}$$

$$\frac{}{(\texttt{Case}\ (\texttt{Inj}_1\ v_1)\ (x_1\ \texttt{=>}\ e_1)\ (x_2\ \texttt{=>}\ e_2)) \mapsto_R [v_1/x_1]e_1}\ \text{red-case1}$$

$$\frac{}{(\texttt{Case}\ (\texttt{Inj}_2\ v_2)\ (x_1\ \texttt{=>}\ e_1)\ (x_2\ \texttt{=>}\ e_2)) \mapsto_R [v_2/x_2]e_2}\ \text{red-case2}$$

$\boxed{e \mapsto e'}$ Expression e takes one step to e'

$$\frac{e \mapsto_R e'}{\mathcal{C}[e] \mapsto \mathcal{C}[e']}\ \text{step-context}$$

Question 3(a). *Progress* says that

> If $\emptyset \vdash e : S$ then either (1) e is a value, or (2) there exists e' such that $e \mapsto e'$.

For most languages, including ours, it is impossible to prove progress without first proving a lemma known as *canonical forms* or *value inversion*.

The first name, canonical forms, comes from the idea that the values of a given type—as opposed to expressions that are not values—are the original or canonical forms of that type. For example, while `(+ 1 1)` and `(- 5 3)` are both *expressions* of type int—and, in a sense, represent the same integer 2 since they all eventually step to 2—we would not consider these expressions as defining the set of integers. But we can say that the *values* of type int—which are the integer constants n—define the integers.

The second name, value inversion, comes from the fact that the lemma uses inversion on a given derivation—but not the inversion we have often used, where we reason either from (a) knowing that we have an expression e of a particular form, say `(Call` e_1 e_2`)`, or (b) knowing that the conclusion of a derivation is by some particular rule, say →Elim. Instead, the inversion is based on the combination of two facts:

- We know that the expression is a value.
- We know something about the expression's type.

Here is the complete value inversion, or canonical forms, lemma for our current language. There is one part for each production in the grammar of types.

Lemma 1 (Value Inversion)**.**

1. *If* $\emptyset \vdash v : \mathsf{unit}$ *then* $v =$ `()`.
2. *If* $\emptyset \vdash v : \mathsf{bool}$ *then either* $v =$ `True` *or* $v =$ `False`.
3. *If* $\emptyset \vdash v : \mathsf{int}$ *then there exists* n *such that* $v = n$.
4. *If* $\emptyset \vdash v : (S_1 \times S_2)$
   then there exist v_1 *and* v_2 *such that* $v =$ `(Pair` v_1 v_2`)` *and* $\emptyset \vdash v_1 : S_1$ *and* $\emptyset \vdash v_2 : S_2$.
5. *If* $\emptyset \vdash v : (S_1 \to S_2)$ *then there exist* x *and* e *such that* $v =$ `(Lam` x e`)` *and* $x : S_1 \vdash e : S_2$.
6. *If* $\emptyset \vdash v : (S_1 + S_2)$ *then either (1) there exists* v_1 *such that* $v =$ `(Inj`$_1$ v_1`)` *and* $\emptyset \vdash v_1 : S_1$ *or (2) there exists* v_2 *such that* $v =$ `(Inj`$_2$ v_2`)` *and* $\emptyset \vdash v_2 : S_2$.

Proof. [Unusually, this proof does not need induction.]

Part 1: The only rule whose conclusion can be $\emptyset \vdash$ `()` : unit is unitIntro. By inversion on unitIntro, we have $v =$ `()`. [In full detail for the impossible cases:

- In type-assum, we have x (which is a value) but the context is $\emptyset$, so the premise is $(x : \mathsf{unit}) \in \emptyset$ which is impossible.
- In →Intro, the expression being typed is a value, but the type is a → which does not match the given unit, so this case is impossible.

- In $\rightarrow$Elim, the expression being typed has the form $(\texttt{Call}\ e_1\ e_2)$, which is not a value.
- In type-true, type-false and intIntro, the expression being typed is a value but the type does not match.
- In type-add, type-sub, type-abs, type-equals, type-lt, type-ite, $\times$Elim1 and $\times$Elim2, the expression being typed is not a value.
- In $\times$Intro, the expression being typed could be a value (if the two subexpressions e_1 and e_2 are values, then $(\texttt{Pair}\ e_1\ e_2)$ is a value), but the type does not match.

End of the detail for the impossible cases.]

Parts 2, 3, 4, 5: [proof omitted]

Part 6: **Question 3(a).** Fill in the four listed cases.

Consider cases of the rule concluding $\emptyset \vdash v : (S_1+S_2)$. [Either explain why the case is impossible, *even if you are only repeating what I gave for Part 1*, or prove the goal for Part 6:
"either (1) there exists v_1 such that $v = (\texttt{Inj}_1\ v_1)$ and $\emptyset \vdash v_1 : S_1$ or (2) there exists v_2 such that $v = (\texttt{Inj}_2\ v_2)$ and $\emptyset \vdash v_2 : S_2$."

- type-assum:

$\emptyset \vdash v : (S_1 + S_2)$	Given
$\Gamma \vdash x : S$	By inversion
$\emptyset \vdash x : (S_1 + S_2)$	By above equation

  However, the premise then becomes $x : (S_1 + S_2) \in \emptyset$, which is impossible.

- $\rightarrow$Intro:

$\emptyset \vdash v : (S_1 + S_2)$	Given
$\Gamma \vdash (\texttt{Lam}\,x\,e) : (S \rightarrow T)$	By inversion

  However, by above equation we have $(S_1 + S_2) = (S \rightarrow T)$, which is impossible.

- $+$Elim:

$\emptyset \vdash v : (S_1 + S_2)$	Given
$\Gamma \vdash (\texttt{Case}\ e\ (x_1 \texttt{=>}\ e_1)\ (x_2 \texttt{=>}\ e_2)) : T$	By inversion
$(\texttt{Case}\ e\ (x_1 \texttt{=>}\ e_1)\ (x_2 \texttt{=>}\ e_2)) = v$	By above equation

  However, the expression has the form $(\texttt{Case}\ e\ (x_1 \texttt{=>}\ e_1)\ (x_2 \texttt{=>}\ e_2))$, which is not possibly a value.

- $+$Intro1:

$\emptyset \vdash v : (S_1 + S_2)$	Given
$v = (\texttt{Inj}_1\ e_1)$	By inversion on $+$Intro1
$e_1 = v_1$	By inversion on value grammar
$v = (\texttt{Inj}_1\ v_1)$	By above equations
$\emptyset \vdash v_1 : S_1$	By inversion on $+$Intro1

- +Intro2: Similar to the +Intro1 case.
- The remaining cases are impossible for reasons similar to those given in Part 1. □