Decomposition Lemma

Lemma (Decomposition)

If \( e : T \) then either \( e \) is a value or there is an \( E \) and \( r \) where \( e = E[r] \) and \( E : S \Rightarrow T \) and \( r : S \) and \( r \) is a redex.

Proof by rule induction on \( e : T \).

Case (1) \( 0 : \text{nat} \): 0 is a value.

Case (5) \( \frac{e_1 : \text{nat}}{\text{succ } e_1 : \text{nat}} \):

Subcase (5a): Suppose \( e_1 \) is a value. From \( e_1 : \text{nat} \) we know that \( e_1 \) is a numerical value by the canonical forms lemma. Therefore \( \text{succ } e_1 \) is a value.

Subcase (5a): Suppose \( e_1 \) is not a value. From the induction hypothesis there is an \( E_1 \) and \( r \) such that \( e_1 = E_1[r] \), \( E_1 : S \Rightarrow \text{nat} \), \( r : S \), and \( r \) is a redex. Then we let \( E = \text{succ } E_1 \) so \( e = E[r] \) and \( E : S \Rightarrow \text{nat} \) and we use the same \( r \) to conclude.
Decomposition Lemma, continued

Case (4)  

\[ \begin{array}{ccc}
  e_1 : \text{bool} & e_2 : T & e_3 : T \\
  \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : T \\
\end{array} \]

Subcase (4a): Suppose \( e_1 \) is a value. Then by the canonical forms lemma, \( e_1 \) is either true or false.

Subsubcase (4ai): Suppose \( e_1 = \text{true} \). Then let \( E = [] \) and \( r = e \).
So we have \( e = E[r] \), \( E : T \Rightarrow T \), \( r : T \), and \( r \rightarrow e_2 \).
Subsubcase (4ai): Suppose \( e_1 = \text{false} \). Same as (4ai) except \( r \rightarrow e_3 \).

Subcase (4b): Suppose \( e_1 \) is not a value. From the induction hypothesis there is an \( E_1 \) and \( r \) such that \( e_1 = E_1[r] \), \( E_1 : S \Rightarrow \text{bool} \), \( r : S \), and \( r \) is a redex. Then we let \( E = \text{if } E_1 \text{ then } e_2 \text{ else } e_3 \) so \( e = E[r] \) and \( E : S \Rightarrow T \) and we use the same \( r \) to conclude.
Lemma (Subject Reduction)

If \( e : T \) and \( e \rightarrow e' \) then \( e' : T \).

Proof by case analysis on \( e : T \).

Case (1) \( 0 : \text{nat} \): There is no \( e' \) such that \( 0 \rightarrow e' \).

Case (6) \( \frac{e_1 : \text{nat}}{\text{pred } e_1 : \text{nat}} \):

Proof by case analysis on \( e \rightarrow e' \).

Subcase (6a): \( \text{pred } 0 \rightarrow 0 \). So \( e' = 0 \) and \( 0 : \text{nat} \).

Subcase (6b): \( \text{pred succ } nv \rightarrow nv \). So \( e' = nv \) and \( nv : \text{nat} \).
Lemma (Replacement)

If $E : S \Rightarrow T$ and $e : S$ then $E[e] : T$.

By rule induction on $E : S \Rightarrow T$.

**Case (1)** $[] : T \Rightarrow T$:

So $E = []$ and $e : T$. Since $[][e] = e$ we have $E[e] : T$.

**Case (2)** $E_1 : S \Rightarrow \text{bool } e_1 : T e_2 : T$:

So $E = \text{if } E_1 \text{ then } e_1 \text{ else } e_2 : S \Rightarrow T$.

By the induction hypothesis we have $E_1[e] : \text{bool}$. Therefore $E[e] : T$.

**Case (3)** $E_1 : S \Rightarrow \text{nat}$:

So $E = \text{succ } E_1$. By the induction hypothesis we have $E_1[e] : \text{nat}$. Therefore $E[e] : \text{nat}$.
Lemma (Progress)

If $e : T$ then either $e$ is a value or an evaluation rule applies to $e$ (i.e., $\exists e'. e \rightarrow e'$).

Proof.

From the decomposition lemma, either either $e$ is a value OR there is an $E$ and $r$ where $e = E[r]$ and $E : S \Rightarrow T$ and $r : S$ and $r$ is a redex.

Case (1): Suppose $e$ is a value. Then we are done.

Case (2): Suppose there is an $E$ and $r$ where $e = E[r]$ and $E : S \Rightarrow T$ and $r : S$ and $r$ is a redex. By definition of redex, there is an $r'$ such that $r \rightarrow r'$. Then we have $e \leftrightarrow E[r']$ and we are done.
Lemma (Subterm Typing)

If $e : T$ and $e = E[r]$ then there is an $S$ such that $E : S \Rightarrow T$ and $r : S$.

Proof.

By rule induction on $e : T$.

Case (1) $0 : \text{nat}$:

So $E = []$ and $r = 0$, and we have $E : \text{nat} \Rightarrow \text{nat}$ and $r : \text{nat}$.

Case (5) $\frac{e_1 : \text{nat}}{\text{succ } e_1 : \text{nat}}$:

By case analysis on $E$, $E$ is either $[]$ or $\text{succ } E_1$.

Subcase (5a): $E = []$. Then $r = e$, $E : \text{nat} \Rightarrow \text{nat}$, and $r : \text{nat}$.

Subcase (5b): $e = \text{succ } E_1$. Applying the induction hypothesis, there is an $S$ such that $E_1 : S \Rightarrow \text{nat}$ and $r : S$. So $\text{succ } E_1 : S \Rightarrow \text{nat}$ and we conclude with the same $r$. 
Lemma (Preservation)

If $e : T$ and $e \rightarrow e'$ then $e' : T$.

Proof.

By case analysis on $e \rightarrow e'$. There is just one case:

$$
\begin{array}{c}
  r \rightarrow r' \\
  E[r] \rightarrow E[r']
\end{array}
$$

So $e = E[r]$ and by the subterm typing lemma, there is an $S$ such that $E : S \Rightarrow T$ and $r : S$. Then by subject reduction lemma we have $r' : S$. Then by the replacement lemma we have $E[r'] : T$ and we are done.
The Simply Typed Lambda Calculus (STLC)

\[ e ::= \quad x \mid true \mid false \mid (\lambda x : T. \ e) \mid (e \ e) \]

\[ v ::= \quad true \mid false \mid (\lambda x : T. \ e) \]

\[ T ::= \quad bool \mid T \to T \]

(The constants true and false technically aren’t part of the STLC, but you have to introduce some type other than function types to get off the ground.)

1. \[ \frac{x : T \in \Gamma}{\Gamma \vdash x : T} \]
2. \[ \frac{}{\Gamma \vdash true : bool} \]
3. \[ \frac{}{\Gamma \vdash false : bool} \]
4. \[ \frac{\Gamma, x : T_1 \vdash e : T_2}{\Gamma \vdash (\lambda x : T_1. \ e) : T_1 \to T_2} \]
5. \[ \frac{\Gamma \vdash e_1 : T_{11} \to T_{12} \quad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash (e_1 \ e_2) : T_{12}} \]
STLC: Evaluation

$$((\lambda x : T. \ e) \ v) \longrightarrow [x \mapsto v] e$$

$$E ::= [] \mid (E \ e) \mid (v \ E)$$

$$e \longrightarrow e' \quad \frac{E[e] \longleftarrow E[e']}{E[e] \longleftarrow E[e']}$$
Properties of the STLC Type System

Lemma (Environment Weakening)
If \( \Gamma \vdash e : T \) and \( x \notin \text{dom}(\Gamma) \) then \( \Gamma, x : S \vdash e : T \).

Lemma (Substitution)
If \( \Gamma, x : S \vdash e : T \) and \( \Gamma \vdash e' : S \), then \( \Gamma \vdash [x \mapsto e']e : T \).

Theorem (Type Safety)
If \( \Gamma \vdash e : T \) and \( e \mapsto^{*} e' \) then \( e' \) is not stuck and \( e' : T \).

Proof.
By the same sequence of lemmas as before (decomposition, subterm typing, subject reduction, replacement, progress, and preservation). However, the details of the proofs change, which is left to you.
The Curry-Howard Correspondence

From logic, recall the rule of modus-ponens:

If \((P \text{ implies } Q) \text{ and } P\), then \(Q\).

Compare this to the typing rule for function application:

\[
\begin{align*}
\Gamma \vdash e_1 : T_{11} \rightarrow T_{12} & \quad \Gamma \vdash t_2 : T_{11} \\
\Gamma \vdash (e_1 \ e_2) : T_{12}
\end{align*}
\]

and think: \(T_{11} \approx P, \ T_{12} \approx Q\).

Also, from logic, recall the rule for implication introduction:

If you can prove \(Q\) assuming \(P\), then \(P\) implies \(Q\).

Compare this to the typing rule for \(\lambda\)s:

\[
\begin{align*}
\Gamma, x : T_1 \vdash e : T_2 \\
\Gamma \vdash (\lambda x : T_1. \ e) : T_1 \rightarrow T_2
\end{align*}
\]

and think: \(T_1 \approx P\) and \(T_2 \approx Q\).

So it turns out, *types correspond to propositions and programs correspond to proofs.*