Type Safety

Last time: type system to prevent programs like (1+2) + "Hello"
How do we know we got it right, i.e. well-typed programs
don't have type errors at runtime?

"Type Safety"! "Well-typed programs can't go wrong" - Robin Milner

2 components (theorems we'll prove):
- Progress: If e is well-typed, it's a value or can take a step
- Preservation: If a well-typed exp takes a step, it's still
  well-typed (with the same type)

\[
\begin{align*}
& e^i \gamma \rightarrow e^2 \gamma \rightarrow e^3 \gamma \rightarrow \ldots \rightarrow v \\
& \text{Prog} \quad \text{Pres} \quad \text{Pres}
\end{align*}
\]

Preservation: If \( e : \tau \) and \( e \rightarrow e' \) then \( e' : \tau \)
Pf: By induction on the derivation of \( e \rightarrow e' \)

S-1 Then \( e = \bar{n}_1 + \bar{n}_2 \) and \( e' = \bar{n}_1 + \bar{n}_2 \).
Need to show \( e' : \tau \).
But how do we know what \( \tau \) is?

Inversion: Use inference rules "upside down"
How do we know \( e = \bar{n}_1 + \bar{n}_2 : \tau \)? Must be the
typing rules.
We now have a case for every rule whose
conclusion can match \( \bar{n}_1 + \bar{n}_2 : \tau \).
There's only one: T-3
\[ e_1 : \text{int} \quad e_2 : \text{int} \quad (T-3) \]

\[ e_1 + e_2 : \text{int} \]

So \( e : \text{c} \) is only possible if \( \text{c} = \text{int} \). 
(We also get that \( \overline{\text{r}}_1 : \text{int} \) and \( \overline{\text{n}}_2 : \text{int} \) but we knew that already and also don't really need it.)

Now for the proof:

5-1. Then \( e = \overline{\text{n}}_1 + \overline{\text{n}}_2 \) and \( e' = \overline{\text{n}}_1 + \overline{\text{n}}_2 \).

By inversion on \( T-3 \), \( \text{c} = \text{int} \). By \( T-1 \), \( e' : \text{int} \).

5-2. Then \( e = \overline{s}_1 \text{^} \overline{s}_2 \) and \( e' = \overline{s}_1 \text{^} \overline{s}_2 \).

By inversion on \( T-4 \), \( \text{c} = \text{string} \). By \( T-2 \), \( e' : \text{string} \).

5-3. Then \( e = \overline{1}'' \text{^} \overline{1}'' \text{^} \overline{1}'' \text{^} \overline{1}'' \) and \( e' = 151 \).

By inversion on \( T-3 \), \( \text{c} = \text{int} \). By \( T-1 \), \( 151 : \text{int} \).

5-4. By inversion on \( T-3 \): \( \text{c} = \text{int} \), \( e_1 : \text{int} \), \( e_2 : \text{int} \).

By induction, \( e_1' : \text{int} \). By \( T-3 \), \( e_1 + e_2 : \text{int} \).

5-5. Then \( e = \overline{1}_1 + e_2 \) and \( e' = \overline{1}_1 + e_2' \) and \( e_2' = e_2' \).

By inversion on \( T-3 \): \( \text{c} = \text{int} \), \( \overline{1}_1 : \text{int} \), \( e_2 : \text{int} \).

By induction, \( e_2' : \text{int} \). By \( T-3 \), \( e' : \text{int} \).

5-6. Then \( e = e_1 \text{^} e_2 \) and \( e' = e_1' \text{^} e_2' \) and \( e_1 = e_1' \).

By inversion on \( T-4 \): \( \text{c} = \text{string} \), \( e_1 : \text{string} \), and \( e_2 : \text{string} \).

By induction, \( e_1' : \text{string} \). By \( T-4 \), \( e' : \text{string} \).

5-7. Then \( e = \overline{s}_1 \text{^} e_2 \) and \( e' = \overline{s}_1 \text{^} e_2' \) and \( e_2' = e_2' \).

By inversion on \( T-4 \): \( \text{c} = \text{string} \), \( e_2 : \text{string} \).

By induction, \( e_2' : \text{string} \). By \( T-4 \), \( e' : \text{string} \).

5-8. Then \( e = [e_0] \) and \( e' = [e_0'] \) and \( e_0 \rightarrow e_0' \).

By inversion on \( T-5 \), \( \text{c} = \text{int} \) and \( e_0 : \text{string} \).

By induction, \( e_0' : \text{string} \). By \( T-5 \), \( e' : \text{int} \). \( \Box \)
Lemma: Canonical Forms
1. If \( e \) val and \( e : \text{int} \), then \( e = \overline{n} \) for some \( n \).
2. If \( e \) val and \( e : \text{string} \), then \( e = \text{"s"} \) for some \( s \).

Proof:
1. By "induction" on the derivation of \( e \) val.
   \( V-1: \) Then \( e = \overline{n} \).
   \( V-2: \) Doesn't apply because then \( e : \text{string} \).

2. Similar.

Progress: If \( e : \text{z} \), then \( e \) val or there exists \( e' \), s.t. \( e \Rightarrow e' \).

Proof:
By induction on the derivation of \( e : \text{z} \).
1. Then \( e = \overline{n} \). By \( V-1 \), \( e \) val.
2. Then \( e = \text{"s"} \). By \( V-2 \), \( e \) val.
3. Then \( e = e_1 + e_2 \) and \( e = \text{int} \) and \( e : \text{int} \) and \( e_2 : \text{int} \).
   By induction, \( e_1 \) val or \( e_1 \Rightarrow e_1' \) for some \( e_1' \).
   - \( e_1 \) val. By canonical forms, \( e_1 = \overline{n_1} \) for some \( n_1 \).
   - By induction, \( e_2 \) val or \( e_2 \Rightarrow e_2' \) for some \( e_2' \).
   - \( e_2 \) val. By canonical forms, \( e_2 = \overline{n_2} \) for some \( n_2 \).
     By \( V-1 \), \( e = \overline{n_1 + n_2} \Rightarrow \overline{n_1} + \overline{n_2} \).
   - \( e_2 \Rightarrow e_2' \). By \( V-2 \), \( e = \overline{n_1 + n_2} \Rightarrow \overline{n_1} + \overline{n_2} \).
   - \( e_1 \Rightarrow e_1' \). By \( V-1 \), \( e = e_1 + e_2 \Rightarrow e_1' + e_2 \).
4. Similar to above.
5. Then \( e = e_0 \) and \( e = \text{int} \) and \( e_0 : \text{string} \).
   By induction, \( e_0 \) val or there exists \( e_0' \), s.t. \( e_0 \Rightarrow e_0' \).
   - \( e_0 \) val. By CF, \( e_0 = \text{"s"} \) for some \( s \).
     By \( V-1 \), \( e_0 \) val.
   - \( e_0 \Rightarrow e_0' \). By \( V-2 \), \( e_0 \) val.

\( Q.E.D. \)