# 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)

$$e_1 \overset{:\tau}{} \underset{\text{Prog}}{\mapsto} e_2 \overset{:\tau^{\text{Pres}}}{} \underset{\text{prog}}{\mapsto} e_3 \overset{:\tau^{\text{Pres}}}{} \mapsto \ldots \mapsto v$$

Preservation: If $e:\tau$ and $e \mapsto e'$ then $e':\tau$
Pf: By induction on the derivation of $e \mapsto e'$

S-1) Then $e = \overline{n_1} + \overline{n_2}$ and $e' = \overline{n_1 + n_2}$.
Need to show $e':\tau$.
But how do we know what $\tau$ is?

Inversion: Use inference rules "upside down"
How do we know $e = \overline{n_1} + \overline{n_2} : \tau$? Must be the typing rules.
We now have a case for every rule whose conclusion can match $\overline{n_1} + \overline{n_2} : \tau$.
There's only one: T-3

$$\frac{e_1 : int \qquad e_2 : int}{e_1 + e_2 : int} \ (T\text{-}3)$$

So $e : \tau$ is only possible if $\tau = int$!
(We also get that $\bar{n}_1 : int$ and $\bar{n}_2 : int$ but we knew that already and also don't really need it)

Now for the proof:

S-1 Then $e = \bar{n}_1 + \bar{n}_2$ and $e' = \overline{n_1 + n_2}$.
By inversion on T-3, $\tau = int$. By T-1, $e' : int$.

S-2. Then $e = \text{"}s_1\text{"} \wedge \text{"}s_2\text{"}$ and $e' = \text{"}s_1 s_2\text{"}$.
By inversion on T-4, $\tau = string$. By T-2, $e' : string$.

S-3. Then $e = |\text{"}s\text{"}|$ and $e' = \overline{|s|}$. By inversion on T-5, $\tau = int$.
By T-1, $\overline{|s|} : int$.

S-4. By inversion on T-3: $\tau = int$, $e_1 : int$, $e_2 : int$
By induction, $e_1' : int$. By T-3, $e_1' + e_2 : int$.

S-5. Then $e = \bar{n}_1 + e_2$ and $e' = \bar{n}_1 + e_2'$ and $e_2 \mapsto e_2'$.
By inversion on T-3: $\tau = int$, $\bar{n}_1 : int$, $e_2 : int$.
By induction, $e_2' : int$. By T-3, $e' : int$.

S-6. Then $e = e_1 \wedge e_2$ and $e' = e_1' \wedge e_2$ and $e_1 \mapsto e_1'$.
By inversion on T-4: $\tau = string$, $e_1 : string$, and $e_2 : string$.
By induction, $e_1' : string$. By T-4, $e' : string$.

S-7. Then $e = \text{"}s_1\text{"} \wedge e_2$ and $e' = \text{"}s_1\text{"} \wedge e_2'$ and $e_2 \mapsto e_2'$.
By inversion on T-4: $\tau = string$, $e_2 : string$.
By induction, $e_2' : string$. By T-4, $e' : string$.

S-8. Then $e = |e_0|$ and $e' = |e_0'|$ and $e_0 \mapsto e_0'$.
By inversion on T-5, $\tau = int$ and $e_0 : string$.
By induction, $e_0' : string$. By T-5, $e' : int$. □

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 $e = \text{"}s\text{"}$ for some $s$.

Pf: 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 $e : \tau$, then $e$ val or there exists $e'$ s.t. $e \mapsto e'$.

Pf: By induction on the derivation of $e : \tau$.

T-1. Then $e = \overline{n}$. By V-1, $e$ val.

T-2. Then $e = \text{"}s\text{"}$. By V-2, $e$ val.

T-3. Then $e = e_1 + e_2$ and $\tau = \text{int}$ and $e_1 : \text{int}$ and $e_2 : \text{int}$.

By induction, $e_1$ val or $e_1 \mapsto e_1'$ 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 $e_2 \mapsto e_2'$ for some $e_2'$.
  - $e_2$ val. By canonical forms, $e_2 = \overline{n_2}$ for some $n_2$.
    By S-1, $e = \overline{n_1} + \overline{n_2} \mapsto \overline{n_1 + n_2}$ ✓
  - $e_2 \mapsto e_2'$. By S-5, $e = \overline{n_1} + e_2 \mapsto \overline{n_1} + e_2'$ ✓
- $e_1 \mapsto e_1'$. By S-4, $e = e_1 + e_2 \mapsto e_1' + e_2$.

T-4. Similar to above.

T-5. Then $e = |e_0|$ and $\tau = \text{int}$ and $e_0 : \text{string}$.

By induction, $e_0$ val or there exists $e_0'$ s.t. $e_0 \mapsto e_0'$.

- $e_0$ val. By CF, $e_0 = \text{"}s\text{"}$ for some $s$.
  By S-3, $|\text{"}s\text{"}| \mapsto \overline{|s|}$.
- $e_0 \mapsto e_0'$. By S-8, $|e_0| \mapsto |e_0'|$

□