Properties of the Typing Relation

Type Safety

The safety (or soundness) of this type system can be expressed by two properties:

1. *Progress:* A well-typed term is not stuck

   If $t : T$, then either t is a value or else $t \longrightarrow t'$ for some t'.

2. *Preservation:* Types are preserved by one-step evaluation

   If $t : T$ and $t \longrightarrow t'$, then $t' : T$.

Inversion

Lemma:

1. If $\mathtt{true} : \mathtt{R}$, then $\mathtt{R} = \mathtt{Bool}$.
2. If $\mathtt{false} : \mathtt{R}$, then $\mathtt{R} = \mathtt{Bool}$.
3. If $\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 : \mathtt{R}$, then $\mathtt{t}_1 : \mathtt{Bool}$, $\mathtt{t}_2 : \mathtt{R}$, and $\mathtt{t}_3 : \mathtt{R}$.
4. If $\mathtt{0} : \mathtt{R}$, then $\mathtt{R} = \mathtt{Nat}$.
5. If $\mathtt{succ}\ \mathtt{t}_1 : \mathtt{R}$, then $\mathtt{R} = \mathtt{Nat}$ and $\mathtt{t}_1 : \mathtt{Nat}$.
6. If $\mathtt{pred}\ \mathtt{t}_1 : \mathtt{R}$, then $\mathtt{R} = \mathtt{Nat}$ and $\mathtt{t}_1 : \mathtt{Nat}$.
7. If $\mathtt{iszero}\ \mathtt{t}_1 : \mathtt{R}$, then $\mathtt{R} = \mathtt{Bool}$ and $\mathtt{t}_1 : \mathtt{Nat}$.

Inversion

Lemma:

1. If $\texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\texttt{t}_1 : \texttt{Bool}$, $\texttt{t}_2 : \texttt{R}$, and $\texttt{t}_3 : \texttt{R}$.
4. If $\texttt{0} : \texttt{R}$, then $\texttt{R} = \texttt{Nat}$.
5. If $\texttt{succ } \texttt{t}_1 : \texttt{R}$, then $\texttt{R} = \texttt{Nat}$ and $\texttt{t}_1 : \texttt{Nat}$.
6. If $\texttt{pred } \texttt{t}_1 : \texttt{R}$, then $\texttt{R} = \texttt{Nat}$ and $\texttt{t}_1 : \texttt{Nat}$.
7. If $\texttt{iszero } \texttt{t}_1 : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$ and $\texttt{t}_1 : \texttt{Nat}$.

Proof: ...

Inversion

Lemma:

1. If $\mathtt{true} : \mathtt{R}$, then $\mathtt{R} = \mathtt{Bool}$.
2. If $\mathtt{false} : \mathtt{R}$, then $\mathtt{R} = \mathtt{Bool}$.
3. If $\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 : \mathtt{R}$, then $\mathtt{t}_1 : \mathtt{Bool}$, $\mathtt{t}_2 : \mathtt{R}$, and $\mathtt{t}_3 : \mathtt{R}$.
4. If $\mathtt{0} : \mathtt{R}$, then $\mathtt{R} = \mathtt{Nat}$.
5. If $\mathtt{succ}\ \mathtt{t}_1 : \mathtt{R}$, then $\mathtt{R} = \mathtt{Nat}$ and $\mathtt{t}_1 : \mathtt{Nat}$.
6. If $\mathtt{pred}\ \mathtt{t}_1 : \mathtt{R}$, then $\mathtt{R} = \mathtt{Nat}$ and $\mathtt{t}_1 : \mathtt{Nat}$.
7. If $\mathtt{iszero}\ \mathtt{t}_1 : \mathtt{R}$, then $\mathtt{R} = \mathtt{Bool}$ and $\mathtt{t}_1 : \mathtt{Nat}$.

Proof: ...

This leads directly to a recursive algorithm for calculating the type of a term...

Typechecking Algorithm

```
typeof(t) = if t = true then Bool
            else if t = false then Bool
            else if t = if t1 then t2 else t3 then
              let T1 = typeof(t1) in
              let T2 = typeof(t2) in
              let T3 = typeof(t3) in
              if T1 = Bool and T2=T3 then T2
              else "not typable"
            else if t = 0 then Nat
            else if t = succ t1 then
              let T1 = typeof(t1) in
              if T1 = Nat then Nat else "not typable"
            else if t = pred t1 then
              let T1 = typeof(t1) in
              if T1 = Nat then Nat else "not typable"
            else if t = iszero t1 then
              let T1 = typeof(t1) in
              if T1 = Nat then Bool else "not typable"
```

Canonical Forms

Lemma:

1. If `v` is a value of type `Bool`, then `v` is either `true` or `false`.
2. If `v` is a value of type `Nat`, then `v` is a numeric value.

Proof:

Canonical Forms

Lemma:

1. If `v` is a value of type `Bool`, then `v` is either `true` or `false`.
2. If `v` is a value of type `Nat`, then `v` is a numeric value.

Proof: Recall the syntax of values:

`v ::=`		*values*
	`true`	*true value*
	`false`	*false value*
	`nv`	*numeric value*
`nv ::=`		*numeric values*
	`0`	*zero value*
	`succ nv`	*successor value*

For part 1,

Canonical Forms

Lemma:

1. If `v` is a value of type `Bool`, then `v` is either `true` or `false`.
2. If `v` is a value of type `Nat`, then `v` is a numeric value.

Proof: Recall the syntax of values:

`v ::=`		*values*
	`true`	*true value*
	`false`	*false value*
	`nv`	*numeric value*
`nv ::=`		*numeric values*
	`0`	*zero value*
	`succ nv`	*successor value*

For part 1, if `v` is `true` or `false`, the result is immediate.

Canonical Forms

Lemma:

1. If `v` is a value of type `Bool`, then `v` is either `true` or `false`.
2. If `v` is a value of type `Nat`, then `v` is a numeric value.

Proof: Recall the syntax of values:

`v ::=`		*values*
	`true`	*true value*
	`false`	*false value*
	`nv`	*numeric value*
`nv ::=`		*numeric values*
	`0`	*zero value*
	`succ nv`	*successor value*

For part 1, if `v` is `true` or `false`, the result is immediate. But `v` cannot be `0` or `succ nv`, since the inversion lemma tells us that `v` would then have type `Nat`, not `Bool`.

Canonical Forms

Lemma:

1. If `v` is a value of type `Bool`, then `v` is either `true` or `false`.
2. If `v` is a value of type `Nat`, then `v` is a numeric value.

Proof: Recall the syntax of values:

`v ::=`		*values*
	`true`	*true value*
	`false`	*false value*
	`nv`	*numeric value*
`nv ::=`		*numeric values*
	`0`	*zero value*
	`succ nv`	*successor value*

For part 1, if `v` is `true` or `false`, the result is immediate. But `v` cannot be `0` or `succ nv`, since the inversion lemma tells us that `v` would then have type `Nat`, not `Bool`. Part 2 is similar.

Progress

Theorem: Suppose $\mathtt{t}$ is a well-typed term (that is, $\mathtt{t} : \mathtt{T}$ for some type $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Progress

Theorem: Suppose $\mathtt{t}$ is a well-typed term (that is, $\mathtt{t} : \mathtt{T}$ for some type $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof:

Progress

Theorem: Suppose $\mathtt{t}$ is a well-typed term (that is, $\mathtt{t} : \mathtt{T}$ for some type $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on a derivation of $\mathtt{t} : \mathtt{T}$.

Progress

Theorem: Suppose $\mathtt{t}$ is a well-typed term (that is, $\mathtt{t} : \mathtt{T}$ for some type $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on a derivation of $\mathtt{t} : \mathtt{T}$.

The T-True, T-False, and T-Zero cases are immediate, since $\mathtt{t}$ in these cases is a value.

Progress

Theorem: Suppose $\mathtt{t}$ is a well-typed term (that is, $\mathtt{t : T}$ for some type $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on a derivation of $\mathtt{t : T}$.

The T-True, T-False, and T-Zero cases are immediate, since $\mathtt{t}$ in these cases is a value.

Case T-If: $\mathtt{t} = \mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3$

$\mathtt{t}_1 : \mathtt{Bool} \qquad \mathtt{t}_2 : \mathtt{T} \qquad \mathtt{t}_3 : \mathtt{T}$

Progress

Theorem: Suppose $\mathtt{t}$ is a well-typed term (that is, $\mathtt{t : T}$ for some type $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on a derivation of $\mathtt{t : T}$.

The T-True, T-False, and T-Zero cases are immediate, since $\mathtt{t}$ in these cases is a value.

Case T-If: $\quad \mathtt{t} = \mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3$

$\mathtt{t}_1 : \mathtt{Bool} \qquad \mathtt{t}_2 : \mathtt{T} \qquad \mathtt{t}_3 : \mathtt{T}$

By the induction hypothesis, either $\mathtt{t}_1$ is a value or else there is some $\mathtt{t}_1'$ such that $\mathtt{t}_1 \longrightarrow \mathtt{t}_1'$. If $\mathtt{t}_1$ is a value, then the canonical forms lemma tells us that it must be either $\mathtt{true}$ or $\mathtt{false}$, in which case either E-IfTrue or E-IfFalse applies to $\mathtt{t}$. On the other hand, if $\mathtt{t}_1 \longrightarrow \mathtt{t}_1'$, then, by E-If, $\mathtt{t} \longrightarrow \mathtt{if}\ \mathtt{t}_1'\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3$.

Progress

Theorem: Suppose $\mathtt{t}$ is a well-typed term (that is, $\mathtt{t} : \mathtt{T}$ for some type $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on a derivation of $\mathtt{t} : \mathtt{T}$.

The cases for rules T-Zero, T-Succ, T-Pred, and T-IsZero are similar.

(Recommended: Try to reconstruct them.)

Preservation

Theorem: If $\mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$, then $\mathtt{t}' : \mathtt{T}$.

Preservation

Theorem: If $\mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$, then $\mathtt{t}' : \mathtt{T}$.

Proof: By induction on the given typing derivation.

Preservation

Theorem: If $\mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$, then $\mathtt{t}' : \mathtt{T}$.

Proof: By induction on the given typing derivation.

Case T-TRUE: $\mathtt{t} = \mathtt{true}$ $\quad$ $\mathtt{T} = \mathtt{Bool}$

Then $\mathtt{t}$ is a value, so it cannot be that $\mathtt{t} \longrightarrow \mathtt{t}'$ for any $\mathtt{t}'$, and the theorem is vacuously true.

Preservation

Theorem: If $\mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$, then $\mathtt{t}' : \mathtt{T}$.

Proof: By induction on the given typing derivation.

Case T-If:

$$\mathtt{t} = \mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \quad \mathtt{t}_1 : \mathtt{Bool} \quad \mathtt{t}_2 : \mathtt{T} \quad \mathtt{t}_3 : \mathtt{T}$$

There are three evaluation rules by which $\mathtt{t} \longrightarrow \mathtt{t}'$ can be derived: E-IfTrue, E-IfFalse, and E-If. Consider each case separately.

Preservation

Theorem: If $\mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$, then $\mathtt{t}' : \mathtt{T}$.

Proof: By induction on the given typing derivation.

Case T-IF:

$$\mathtt{t} = \mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \quad \mathtt{t}_1 : \mathtt{Bool} \quad \mathtt{t}_2 : \mathtt{T} \quad \mathtt{t}_3 : \mathtt{T}$$

There are three evaluation rules by which $\mathtt{t} \longrightarrow \mathtt{t}'$ can be derived: E-IFTRUE, E-IFFALSE, and E-IF. Consider each case separately.

Subcase E-IFTRUE: $\quad \mathtt{t}_1 = \mathtt{true} \qquad \mathtt{t}' = \mathtt{t}_2$

Immediate, by the assumption $\mathtt{t}_2 : \mathtt{T}$.

(E-IFFALSE subcase: Similar.)

Preservation

Theorem: If $\mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$, then $\mathtt{t}' : \mathtt{T}$.

Proof: By induction on the given typing derivation.

Case T-If:

$$\mathtt{t} = \mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \quad \mathtt{t}_1 : \mathtt{Bool} \quad \mathtt{t}_2 : \mathtt{T} \quad \mathtt{t}_3 : \mathtt{T}$$

There are three evaluation rules by which $\mathtt{t} \longrightarrow \mathtt{t}'$ can be derived: E-IfTrue, E-IfFalse, and E-If. Consider each case separately.

Subcase E-If: $\quad \mathtt{t}_1 \longrightarrow \mathtt{t}_1' \qquad \mathtt{t}' = \mathtt{if}\ \mathtt{t}_1'\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3$

Applying the IH to the subderivation of $\mathtt{t}_1 : \mathtt{Bool}$ yields $\mathtt{t}_1' : \mathtt{Bool}$. Combining this with the assumptions that $\mathtt{t}_2 : \mathtt{T}$ and $\mathtt{t}_3 : \mathtt{T}$, we can apply rule T-If to conclude that $\mathtt{if}\ \mathtt{t}_1'\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 : \mathtt{T}$, that is, $\mathtt{t}' : \mathtt{T}$.