Harry Langford **CST Part II: Types, SV 1**
hjel2@cam.ac.uk 2023-11-11 16:00, Virtual

1 Warmup Questions

1. for a strongly-typed programming language, what are the advantages of a type system?
   - abstraction
   - optimization
   - safety
2. Describe the concepts of weakening, exchange and substitution.
   - Weakening

     Generally: if a term T has property P under context Θ, Θ' then T will still have property P in the context Θ, α, Θ'.

     Concretely, weakening for Simply Typed Lambda Calculus (STLC) is defined as:

     $$\Theta, \Theta' \vdash e : A \implies \Theta, x : X, \Theta' \vdash e : A$$
   - Exchange

     Generally: if a term T has property P under context $\Theta, \alpha, \beta, \Theta'$ then T will still have property P in the context $\Theta, \beta, \alpha, \Theta'$. Intuitively, exchange means that the order in which objects occur in the context is not important.

     Concretely, exchange for the STLC is defined as:

     $$\Theta, x : X, y : X, \Theta' \vdash e : A \implies \Theta, y : Y, x : X, \Theta' \vdash e : A$$
   - Substitution

     Concretely, substitution for the STLC is defined as:

     $$\Theta, x : X \vdash e : A \wedge \Theta \vdash e' : X \implies \Theta \vdash [e'/x]e : A$$
3. Define the two properties of type safety.

   The two properties of Type Safety are Progress and Type Preservation.

   Definition 1 (Progress)**.** If an expression e is well-typed then it is either a value, or there exists some expression e' such that $e \rightsquigarrow e'$.

   Definition 2 (Type Preservation)**.** If an expression e has the type T and $e \rightsquigarrow e'$ then e' also has they type T.
4. What is the Curry-Howard Correspondence?

   The Curry Howard Correspondence is a correspondence between types systems an proof systems. The exact correspondences are in Table 1.

Type Systems	Proof Systems
unit type	truth
null type	falsehood
functions	implication
sum type	logical or
product type	logical and
reduction	normalisation
value	cut-free proof
reduction order	normalisation order

Table 1: Correspondences in the Curry Howard Correspendence

5. Give the typing rules and operational semantics of TLC for functions, pairs and sums.

   Typing Rules:

$$\frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash \lambda x : A.\, e : A \to B} \qquad \frac{\Gamma \vdash e : A \to B \qquad \Gamma \vdash e' : A}{\Gamma \vdash e\, e' : B}$$

$$\frac{\Gamma \vdash e_1 : A \qquad \Gamma \vdash e_2 : B}{\Gamma \vdash (e_1, e_2) : A \times B} \qquad \frac{\Gamma \vdash e : A \times B}{\Gamma \vdash \text{fst}\, e : A} \qquad \frac{\Gamma \vdash e : A \times B}{\Gamma \vdash \text{snd}\, e : B}$$

$$\frac{\Gamma \vdash e_1 : A}{\Gamma \vdash L\, e_1 : A + B} \qquad \frac{\Gamma \vdash e_2 : B}{\Gamma \vdash R\, e_1 + e_2 : A + B} \qquad \frac{\Gamma \vdash e : A + B \qquad \Gamma, x : A \vdash e_1 : C \qquad \Gamma, y : B \vdash e_2 : C}{\Gamma \vdash \text{case}(e, L\, x \to e_1, R\, y \to e_2) : C}$$

   Operational Semantics:

$$\frac{e \rightsquigarrow e'}{e\, e'' \rightsquigarrow e'\, e''} \qquad \frac{e' \rightsquigarrow e''}{v\, e' \rightsquigarrow v\, e''} \qquad \frac{}{(\lambda x.\, e)\, v \rightsquigarrow [v/x]e}$$

$$\frac{e_1 \rightsquigarrow e_1'}{(e_1, e_2) \rightsquigarrow (e_1', e_2)} \qquad \frac{e_2 \rightsquigarrow e_2'}{(v_1, e_2) \rightsquigarrow (v_1, e_2')} \qquad \frac{e \rightsquigarrow e'}{\text{fst}\, e \rightsquigarrow e'} \qquad \frac{e \rightsquigarrow e'}{\text{snd}\, e \rightsquigarrow e'}$$

$$\frac{e \rightsquigarrow e'}{L\, e \rightsquigarrow L\, e'} \qquad \frac{e \rightsquigarrow e'}{R\, e \rightsquigarrow L\, e'} \qquad \frac{e \rightsquigarrow e'}{\text{case}(e', L\, x \to e_1, R\, y \to e_2)}$$

2 Regular Questions

1. Extend the type preservation and progress proof to cover AND.

$$\Phi(e, \tau) \triangleq e : \tau \implies e \text{ is a value} \vee \exists e'. e \rightsquigarrow e'$$

   Prove Progress by structural induction on the derivation of $e : \tau$. Let $\Phi(e, \tau)$ be the induction hypothesis:

   Case (AND):

$\dfrac{e_1 : \text{bool} \qquad e_2 : \text{bool}}{e_1 \wedge e_2 : \text{bool}}$	by assumption

   Case e_1 is not a value

$e_1 \rightsquigarrow e_1'$	by induction hypothesis
$e_1 \wedge e_2 \rightsquigarrow e_1' \wedge e_2$	by the reduction rules

   Case e_1 is a value

   Case e_2 is not a value

$e_2 \rightsquigarrow e_2'$	by induction hypothesis
$e_1 \wedge e_2 \rightsquigarrow e_1 \wedge e_2'$	by the reduction rules

Case e_2 is a value

$e_1 \wedge e_2$	is a value

$$\Phi(e, e', \tau) \triangleq e : \tau \wedge e \rightsquigarrow e' \implies e' : \tau$$

Prove Type Preservation by structural induction on the derivation $e : \tau$. Let $\Phi(e, e', \tau)$ be the induction hypothesis:

Case (AND):

$e \rightsquigarrow e'$	by assumption
$\dfrac{e_1 : \text{bool} \qquad e_2 : \text{bool}}{e_1 \wedge e_2 : \text{bool}}$	by assumption
$e = e_1 \wedge e_2$	by assumption

Case e_1 is not a value

$e_1 \rightsquigarrow e_1'$	by Progress
$e_1' : \text{bool}$	by induction hypothesis
$e_1 \wedge e_2 \rightsquigarrow e_1' \wedge e_2$	by the reduction rules
$e_1' \wedge e_2 = e'$	by determinacy
$e' : \text{bool}$	by (AND)

Case e_1 is a value

Case e_2 is not a value

$e_2 \rightsquigarrow e_2'$	by Progress
$e_2' : \text{bool}$	by induction hypothesis
$e_1 \wedge e_2 \rightsquigarrow e_1 \wedge e_2'$	by the reduction rules
$e_1 \wedge e_2 = e'$	by determinacy
$e' : \text{bool}$	by (AND)

Case e_1 is a value

$e_1 \wedge e_2$ is a value. Thus e is a value and $\nexists e'. e \rightsquigarrow e'$: the premise was false in this case so Φ holds trivially.

2. Prove type safety for the unit and function cases of TLC.

   - Progress

$$\Phi(\Gamma, e, \tau) \triangleq \Gamma \vdash e : \tau \implies e \text{ is a value } \vee \exists e'.\, e \rightsquigarrow e'$$

I will prove this by structural induction on the derivation $\Gamma \vdash e : \tau$ with the induction hypothesis that for all subexpressions e, $\Phi(\Gamma, e, \tau)$. I will prove only the unit and function cases.

- **Case unit**

  By assumption, we have:

$$\frac{}{\Gamma \vdash \langle\rangle : 1}$$

  We have that $\langle\rangle$ is a value. Thus we have Φ in this case.

- **Case abstr**

  By assumption, we have:

$$\frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash \lambda x : A.\, e : A \to B}$$

  $\lambda x.\, e$ is a value. Thus we have Φ in this case.

- **Case app**

  By assumption, we have:

$$\frac{\Gamma \vdash e_1 : A \to B \qquad \Gamma \vdash e_2 : A}{\Gamma \vdash e_1\, e_2 : B}$$

$\Gamma \vdash e_1 : A \to B$	subderivation
$\Gamma \vdash e_2 : A$	subderivation

  By the induction hypothesis, we have that e_1 is a value or $\exists e_1'.\, e_1 \rightsquigarrow e_1'$

  Case $\exists e_1'.\, e_1 \rightsquigarrow e_1'$

$$\frac{e_1 \rightsquigarrow e_1'}{e_1\, e_2 \rightsquigarrow e_1'\, e_2}$$

  Thus we have that $e \rightsquigarrow e_1'\, e_2$

  Case e_1 is a value

  By the induction hypothesis, we have that e_2 is a value or $\exists e_2'.\, e_2 \rightsquigarrow e_2'$

  Case $\exists e_2'.\, e_2 \rightsquigarrow e_2'$

$$\frac{e_2 \rightsquigarrow e_2'}{e_1\, e_2 \rightsquigarrow e_1\, e_2'}$$

  Thus we have that $e \rightsquigarrow e_1\, e_2'$

Case e_2 is a value v_2

For e_1 to be a value and have type $A \to B$, we have that it must be of the form $\lambda x : A.\, e_3$ for some e_3

$$\frac{}{(\lambda x : A.\, e_3)\, v_2 \rightsquigarrow [v_2/x]e_3}$$

Thus we have that $e \rightsquigarrow [v_2/x]e_3$

- Type Preservation

$$\Phi(\Gamma, e, \tau) \triangleq \Gamma \vdash e : \tau \wedge e \rightsquigarrow e' \implies \Gamma \vdash e' : \tau$$

I will prove this by structural induction on the derivation of $e \rightsquigarrow e'$ with the induction hypothesis that for all subexpressions e, $\Phi(\Gamma, e, \tau)$. I will prove only the unit and function cases.

  - **Case Unit**

    There is no reduction rule for any expression of type unit. Therefore, the premise of Φ is false and so it holds trivially.

  - **Case app1**

$\frac{e_1 \rightsquigarrow e_1'}{e_1\, e_2 \rightsquigarrow e_1'\, e_2}$	by assumption
$\Gamma \vdash e_1\, e_2 : \tau$	by assumption
$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1\, e_2 : \tau}$	by inversion
$\Gamma \vdash e_1 : \tau' \to \tau$	subderivation
$\Gamma \vdash e_2 : \tau$	subderivation
$\Gamma \vdash e_1' : \tau' \to \tau$	by induction hypothesis
$\frac{\Gamma \vdash e_1' : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1'\, e_2 : \tau}$	

Thus, we have proved Φ in this case!

  - **Case app2**

$\frac{e_2 \rightsquigarrow e_2'}{v\, e_2 \rightsquigarrow v\, e_2'}$	by assumption
$\Gamma \vdash v\, e_2 : \tau$	by assumption
$\frac{\Gamma \vdash v : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash v\, e_2 : \tau}$	by inversion
$\Gamma \vdash v : \tau' \to \tau$	subderivation
$\Gamma \vdash e_2 : \tau$	subderivation
$\Gamma \vdash e_2' : \tau'$	by induction hypothesis
$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2' : \tau}{\Gamma \vdash e_1\, e_2' : \tau}$	

Thus, we have proved Φ in this case!

– **Case sub**

$\dfrac{}{(\lambda x : \tau'.\, e)\, v \rightsquigarrow [v/x]e}$	by assumption
$\Gamma \vdash (\lambda x : \tau'.\, e)\, v : \tau$	by assumption
$\dfrac{\Gamma \vdash \lambda x : \tau'.\, e : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash v\, e_2 : \tau}$	by inversion
$\Gamma \vdash \lambda x : \tau'.\, e : \tau' \to \tau$	subderivation
$\Gamma \vdash e_2 : \tau$	subderivation
$\dfrac{\Gamma, x : \tau' \vdash e : \tau}{\Gamma \vdash \lambda x.\, e : \tau' \to \tau}$	by inversion
$\Gamma, x : \tau' \vdash e : \tau$	subderivation
$\Gamma \vdash [v/x]e : \tau$	substitution

Thus, we have proved Φ in this case!

3. In the simply typed λ-calculus, is there any context Γ and type Y for which $\Gamma \vdash x\, x : Y$ (where x is a free variable defined in Γ)? If so, give Γ, Y and show a typing derivation, otherwise prove that no such context and type exists.

   No such type exists! Consider the following proof by contradiction.

   Assume that there exists a context Γ and type Y such that $\Gamma \vdash x\, x : Y$. Thus the following proof tree must hold.

$$\dfrac{\dfrac{}{\Gamma \vdash x : X \to Y} \qquad \dfrac{}{\Gamma \vdash x : X}}{\Gamma \vdash x\, x : Y}$$

   However, this implies that both $x : X \to Y \in \Gamma$ and $x : X \in \Gamma$. Since these are not equal types, we have that Γ is an ill-formed typing context. This is a contradiction! Thus there exists no context Γ and type Y such that $\Gamma \vdash x\, x : Y$.

4. Under the Curry-Howard Correspondence (CHC), a proposition P is represented by a type π. Proving P corresponds to providing a term t such that $\cdot \vdash p : \pi$.

   (a) We represent the negation $\neg P$ by a type $\pi \to 0$. why is this a reasonable representation?

   By the CHC, we have that $\pi \to 0$ corresponds to the logical expression $P \supset 0$.

$$P \subset 0 \equiv \neg P \vee 0 \equiv \neg P$$

   Thus, by the CHC, we have that the proposition represented by this type is logically equivalent to the proposition $\neg P$.

   (b) Try to prove De Morgan's laws by converting each propositional formula to a type and providing a term of that type in the empty context.

   Let π be the type corresponding to the propositional formula P, and let ρ be the type corresponding to the propositional formula Q.

   i. $\neg P \wedge \neg Q \supset \neg P \vee Q$

   By the CHC we have that $\neg P \wedge \neg Q \supset \neg P \vee Q$ corresponds to the type $(\pi \to 0) \times (\rho \to 0) \to (\pi + \rho) \to 0$. I will prove that this proposition holds by showing that there exists a term which has that type for arbitrary π, ρ.

   Let $\Gamma = x : (\pi \to 0) \times (\rho \to 0), y : \pi + \rho$

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash : \pi + \rho} \quad \dfrac{\dfrac{\dfrac{}{\Gamma, a : \pi \vdash x : (\pi \to 0) \times (\rho \to 0)}}{\Gamma, a : \pi \vdash \mathrm{fst}\, x : \pi \to 0} \quad \dfrac{}{\Gamma, a : \pi \vdash a : \pi}}{\Gamma, a : \pi \vdash (\mathrm{fst}\, x)\, a : 0} \quad \dfrac{\dfrac{\dfrac{}{\Gamma, a : \pi \vdash x : (\pi \to 0) \times (\rho \to 0)}}{\Gamma, a : \pi \vdash \mathrm{snd}\, x : \rho \to 0} \quad \dfrac{}{\Gamma, b : \pi \vdash b : \rho}}{\Gamma, a : \pi \vdash (\mathrm{fst}\, x)\, a : 0}}{\dfrac{\Gamma \vdash \mathrm{case}(y, L\, a \to (\mathrm{fst}\, x)\, a, R\, b \to (\mathrm{snd}\, x)\, b) : 0}{\dfrac{x : (\pi \to 0) \times (\rho \to 0) \vdash \lambda y : \pi + \rho.\, \mathrm{case}(y, L\, a \to (\mathrm{fst}\, x)\, a, R\, b \to (\mathrm{snd}\, x)\, b) : (\pi + \rho) \to 0}{\cdot \vdash \lambda x : (\pi \to 0) \times (\rho \to 0).\, \lambda y : \pi + \rho.\, \mathrm{case}(y, L\, a \to (\mathrm{fst}\, x)\, a, R\, b \to (\mathrm{snd}\, x)\, b) : (\pi \to 0) \times (\rho \to 0) \to (\pi + \rho) \to 0}}}}{}$$

Figure 1: Typing derivation for a term of type $(\pi \to 0) \times (\rho \to 0) \to (\pi + \rho) \to 0$

ii. $\neg(P \vee Q) \supset \neg P \wedge \neg Q$

By the CHC we have that $\neg P \wedge \neg Q \supset \neg P \vee Q$ corresponds to the type $((\pi + \rho) \to 0) \supset (\pi \to 0) \times (\rho \to 0)$. I will prove that this proposition holds by showing that there exists a term which has that type for arbitrary π, ρ.

Let $\Gamma = x : (\pi + \rho) \to 0$.

iii. $\neg P \vee \neg Q \supset \neg(P \wedge Q)$

By the CHC, we have that the type this proposition corresponds to is $((\pi \to 0) + (\rho \to 0)) \to (\pi \times \rho \to 0)$.

Let $\Gamma = x : (\pi \to 0) + (\rho \to 0), y : \pi \times \rho$

iv. $\neg(P \wedge Q) \supset \neg P \vee \neg Q$

This law does not hold! The type it corresponds to under the CHC is $(\pi \times \rho \to 0) \to (\pi \to 0) \times (\rho \to 0)$. No term can have this type in the empty context. Such a term would take a function $f : (\pi \times \rho) \to 0$ and return functions $g_1 : \pi \to 0$, $g_2 : \rho \to 0$. Either function individually only takes a term of type π or ρ and so cannot call f. Thus the only way such a term could exist is if it could make terms of type 0 – which would imply the language was unsound.

(c) De Morgan's laws all hold in propositional logic. What does 4b tell you about the STLC in relation to propositional logic?

The STLC does not correspond to propositional logic.

5. Extend logical relation to support products and sums.

We define a logical relation Halt_T for all types T:

$$\begin{aligned}
&\text{Halt}_0 = \emptyset\\
&\text{Halt}_1 = \{e \mid e \text{ halts}\}\\
&\text{Halt}_{X \to Y} = \{e \mid e \text{ halts} \wedge \forall e' \in \text{Halt}_X .\, e\ e' \in \text{Halt}_X\}\\
&\text{Halt}_{X \times Y} = \{e \mid e \text{ halts}\}\\
&\text{Halt}_{X+Y} = \{L\, e \mid e \in \text{Halt}_X\} \cup \{R\, e \mid e \in \text{Halt}_Y\}
\end{aligned}$$

6. 3 2023 Paper 8 Question 13

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2023p8q13.pdf

Consider Gödel's T, the Simply-Typed Lambda Calculus with function and natural number types, with zero, successor and iterator term formers for teh natural number type.

(*a*) Define a logical relation suitable for establishing the termination of closed programs in this language.

I define an intermediate logical relation and use this to define a logical relation which is suitable for establishing termination in the language.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, y : \pi \vdash x : (\pi + \rho) \to 0} \quad \dfrac{\dfrac{}{\Gamma, y : \pi \vdash y : \pi}}{\Gamma, y : \pi \vdash L\,y : (\pi + \rho)}}{\Gamma, y : \pi \vdash x\,(L\,y) : 0}}{\Gamma \vdash \lambda y : \pi.\, x\,(L\,y) : (\pi \to 0)} \quad \dfrac{\dfrac{\dfrac{}{\Gamma, y : \rho \vdash x : (\pi + \rho) \to 0} \quad \dfrac{\dfrac{}{\Gamma, y : \rho \vdash y : \rho}}{\Gamma, y : \rho \vdash R\,y : (\pi + \rho)}}{\Gamma, y : \rho \vdash x\,(R\,y) : 0}}{\Gamma \vdash \lambda y : \rho.\, (R\,y) : (\rho \to 0)}}{\Gamma \vdash (\lambda y : \pi.\, x\,(L\,y), \lambda y : \rho.\, x\,(R\,y) : (\pi \to 0) \times (\rho \to 0)}}{\cdot \vdash \lambda x : (\pi + \rho) \to 0.\, (\lambda y : \pi.\, x\,(L\,y), \lambda y : \rho.\, x\,(R\,y) : ((\pi + \rho) \to 0) \to (\pi \to 0) \times (\rho \to 0)}$$

Figure 2: Typing derivation for a term of type $((\pi + \rho) \to 0) \supset (\pi \to 0) \times (\rho \to 0)$

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : (\pi \to 0) + (\rho \to 0)} \quad \dfrac{\dfrac{}{\Gamma, a : \pi \to 0 \vdash a : \pi \to 0} \quad \dfrac{\dfrac{}{\Gamma, a : \pi \to 0 \vdash y : \pi \times \rho}}{\Gamma, a : \pi \to 0 \vdash \mathrm{fst}\, y : \pi}}{\Gamma, a : \pi \to 0 \vdash a\,(\mathrm{fst}\, y) : 0} \quad \dfrac{\dfrac{}{\Gamma, b : \rho \to 0 \vdash b : \rho \to 0} \quad \dfrac{\dfrac{}{\Gamma, b : \rho \to 0 \vdash y : \pi \times \rho}}{\Gamma, b : \rho \to 0 \vdash \mathrm{snd}\, y : \rho}}{\Gamma, b : \rho \to 0 \vdash b\,(\mathrm{snd}\, y) : 0}}{\dfrac{\Gamma \vdash \mathrm{case}(x, L\, a \to a\,(\mathrm{fst}\, y), b \to b\,(\mathrm{snd}\, y)) : 0}{x : (\pi \to 0) + (\rho \to 0) \vdash \lambda y : \pi \times \rho.\, \mathrm{case}(x, L\, a \to a\,(\mathrm{fst}\, y), b \to b\,(\mathrm{snd}\, y)) : \pi \times \rho \to 0}}}{\cdot \vdash \lambda x : (\pi \to 0) + (\rho \to 0).\, \lambda y : \pi \times \rho.\, \mathrm{case}(x, L\, a \to a\,(\mathrm{fst}\, y), b \to b\,(\mathrm{snd}\, y)) : ((\pi \to 0) + (\rho \to 0)) \to (\pi \times \rho \to 0)}$$

Figure 3: Typing derivation for a term of type $((\pi \to 0) + (\rho \to 0)) \to (\pi \times \rho \to 0)$

The definition of the intermediate logical relation is as follows:

$$\begin{aligned}
\mathrm{Halt}_1 =& \{e \mid e \text{ halts}\}\\
\mathrm{Halt}_{\mathbb{N}_{\leq 0}} =& \{e \mid e \rightsquigarrow^* z\}\\
\forall i \in \mathbb{N}.\, \mathrm{Halt}_{\mathbb{N}_{\leq} i+1} =& \{e \mid e \rightsquigarrow^* s(e') \wedge e' \in \mathrm{Halt}_{\mathbb{N} \leq i}\}\\
\mathrm{Halt}_{\mathbb{N}} =& \bigcup_{i \in \mathbb{N}} \mathrm{Halt}_{\mathbb{N}_{\leq} i}\\
\mathrm{Halt}_{X \to Y} =& \{e \mid e \text{ halts} \wedge \forall e' \in \mathrm{Halt}_X .\, e\, e' \in \mathrm{Halt}_Y\}\\
& \cup \{\mathrm{iter}(e, z \to e_1, s(x) \to e_2) \mid e \in \mathrm{Halt}_{\mathbb{N}} \wedge e_1 \in \mathrm{Halt}_{X \to Y}\\
& \quad \wedge \lambda x : X \to Y .\, e_2 \in \mathrm{Halt}_{(X \to Y) \to (X \to Y)}\big\}
\end{aligned}$$

(*b*) State the fundamental lemma for this language.

$$x_1 : X_1, x_2 : X_2, \ldots, x_n : X_n \vdash e : T \wedge \forall i \in \{1, \ldots n\}. \cdot \vdash v_i : X_i \implies [v_1/x_1, \ldots, v_n/x_n]e \in \mathrm{Halt}_T$$

(*c*) State formally what it means for a set of terms X to he "closed under reduction".

The set of terms X is "closed under reduction" if, and only if:

$$\forall e \in X. e \rightsquigarrow e' \implies e' \in X$$

(*d*) Prove the fundamental lemma holds for the iterator case.

Prove the fundamental lemma for Gödel's T by structural induction on $x_1 : X_1, \ldots, x_n : X_n \vdash e : Z$.

For brevity, let $\Gamma = x_1 : X_1, x_2 : X_2, \ldots, x_n : X_n$ and $\gamma = [v_1/x_1, \ldots, v_n/x_n]$

Case (iter):

$\Gamma \vdash e : Z$	by assumption
$e = \mathrm{iter}(e_0, z \to e_1, s(x) \to e_2)$	by assumption
$\dfrac{\Gamma \vdash e_0 : \mathbb{N} \qquad \Gamma : Z \qquad \Gamma, x : Z \vdash e_2 : X}{\Gamma \vdash \mathrm{iter}(e_0, z \to e_1, s(x) \to e_2) : X}$	by assumption
$\Gamma \vdash e_0 : \mathbb{N}$	subderivation
$\Gamma \vdash e_1 : Z$	subderivation
$\Gamma, x : Z \vdash e_2 : Z$	subderivation
$\forall i \in \{1, \ldots, n\}.\, v_i : X_i$	by assumption
γe	
$= \gamma \mathrm{iter}(e_0, z \to e_1, s(x) \to e_2)$	by assumption
$= \mathrm{iter}(\gamma e_0, z \to \gamma e_1, s(x) \to \gamma e_2)$	by definition of substitution
$\gamma e_0 \in \mathrm{Halt}_{\mathbb{N}}$	by the induction hypothesis
$\gamma e_1 \in \mathrm{Halt}_Z$	by the induction hypothesis

Notice that since we have $\gamma e_0 \in \mathrm{Halt}_{\mathbb{N}}$, we have $\gamma e_0 \rightsquigarrow^* n$. I will perform structural induction on n for the remainder of the proof (*i.e.* to prove $\gamma \mathrm{iter}(e_0, z \to e_1, s(x) \to e_2) \in \mathrm{Halt}_Z$)

Case (z)

$\gamma e_0 \rightsquigarrow^* z$	by assumption
$\gamma\text{iter}(e_0, z \to e_1, s(x) \to e_2) \rightsquigarrow \gamma\text{iter}(z, z \to e_1, s(x) \to e_2)$	by the reduction rules
$\gamma\text{iter}(z, z \to e_1, s(x) \to e_2) \rightsquigarrow \gamma e_1$	by the reduction rules
$\gamma e_1 \rightsquigarrow^* v$	by assumption, since $\gamma e_1 \in \text{Halt}_Z$
$\gamma\text{iter}(z, z \to e_1, s(x) \to e_2) \rightsquigarrow^* v$	by transivitity of reduction

Case ($s(v')$)

$\gamma e_0 \rightsquigarrow^* s(x)$	by assumption
$\gamma\text{iter}(e_0, z \to e_1, s(x) \to e_2) \rightsquigarrow \gamma\text{iter}(s(v'), z \to e_1, s(x) \to e_2)$	by the reduction rules
$\gamma\text{iter}(s(v'), z \to e_1, s(x) \to e_2) \rightsquigarrow \gamma[v'/x]e_2$	by the reduction rules
$\gamma[v'/x]e_1 \rightsquigarrow^* v$	by assumption, since $\gamma[v'/x]e_1 \in \text{Halt}_Z$
$\gamma\text{iter}(s(v'), z \to e_1, s(x) \to e_2) \rightsquigarrow^* v$	by transivitity of reduction

We have now proved the fundamental lemma in the iter case!

4 2020 Paper 8 Question 15

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2020p8q15.pdf

(*b*) (i) Give the typing rules for Peano natural numbers and their eliminator.

Peano natural numbers can be represented either as z (for zero) or $s(x)$ for the successor of some other natural number x. Their eliminator is the `iter` function – which can be interpreted as bounded recursion.

$$\frac{}{\Gamma \vdash z : \mathbb{N}} \qquad \frac{\Gamma \vdash x : \mathbb{N}}{\Gamma \vdash s(x) : \mathbb{N}} \qquad \frac{\Gamma \vdash e : \mathbb{N} \qquad \Gamma \vdash e_1 : X \qquad \Gamma, x : X \vdash e_2 : X \to X}{\text{iter}(e, z \to e_1, s(x) \to e_2) : X}$$

(ii) Using the rules given above, define the addition function.

$$\text{add} \triangleq \lambda m.\, \lambda n.\, \text{iter}(m, z \to n, s(x) \to s(x))$$

(iii) Let a binary tree be either a leaf `Leaf` or a node `Node(l, x, r)` where `l` and `r` are subtrees, and `x` is a natural number. Give typing rules for trees corresponding to this prose description, including an eliminator.

I will consider the eliminator fold which aggregates the nodes in a tree with a user-defined ternary operation. For example $\text{Fold}(e, \text{Leaf} \to z, \text{Node}(l, x, r) \to \text{add}(l, \text{add}(x, r)))$ would sum all the nodes in the tree.

$$\frac{}{\Gamma \vdash \text{Leaf} : \text{tree}} \qquad \frac{\Gamma \vdash e_1 : \text{tree} \qquad \Gamma \vdash e_2 : \text{tree}}{\Gamma \vdash \text{Node}(e_1, x, e_2) : \text{tree}}$$

$$\frac{\Gamma \vdash e : \text{tree} \qquad \Gamma \vdash e_1 : X \qquad \Gamma, l : X, x : \mathbb{N}, r : X \vdash e_2 : X}{\Gamma \vdash \text{fold}(e, \text{Leaf} \to e_1, \text{Node} \to e_2) : X}$$

(iv) Using the rules given above, define a function `size` which takes a binary tree and returns the total number of nodes in the tree.

$$\text{size} = \lambda x : \text{tree}.\, \text{fold}(x, \text{Leaf} \to z, \text{Node}(l, x, r) \to s(\text{add}(l, r))$$

5 Extension Questions

1. Find a context Γ under which the term $f\,x\,y$ has type Bool. *i.e.* $\Gamma \vdash f\,x\,y : \text{Bool}$.

   In the STLC the representation of Bool is given by $1 + 1$. I will work in this langauge.

   $$\Gamma = f : 1 \to 1 \to 1 + 1, x : 1, y : 1$$

2. Can you give a simple description of the set of all such contexts?

   $$G = \{(f : X \to Y \to 1 + 1, x : X, y : Y) \mid X, Y \text{ are valid types}\}$$

3. OCaml has a polymorphic equals-function. That is, equal : $a \to a \to$ bool is defined for all types. What are the implications on safety of the type system?

   From computation theory; we have that for the lambda calculus, the existence of an equals function implies that there are expressions which loop infinitely. Applying similar logic here means that the existence of a polymorphic equals function means there is an expression in OCaml which loops infinitely.