Lecture 13: Initiality Theorem for STLC Continued

Lecturer: Max S. New
Scribe: Eric Bond

October 8, 2025

1 Review

Recall the definition of a simply type category with families.

Definition 1.1. A simply typed category with families S consists of:

1. A category S_c with a terminal object
2. A set S_t
3. A family of presheaves $\{Tm(A) : Psh\ S_c\}_{\forall A \in S_t}$ indexed by S_t
4. An operation $\Gamma \times A : Ob\ S_c$ where $\Gamma : Ob\ S_c$ and $A : S_t$ with projections $p_{\Gamma,A} : S_c[\Gamma \times A, \Gamma]$ and $q_{\Gamma,A} : Tm(\Gamma \times A, A)$ such that forall $\gamma : \Delta \to \Gamma$ and $a : Tm(\Delta, A)$ there exists a unique $\langle \gamma, a \rangle : \Delta \to \Gamma \times A$ such that $p_{\Gamma,A} \circ \langle \gamma, a \rangle = \gamma$ and $q_{\Gamma,a}[\langle \gamma, a \rangle] = a$

Alternatively, the condition for the operation $\Gamma \times A$ can be stated as:

$$\forall \Gamma, A.\ \mathcal{Y}(\Gamma \times A) \cong \mathcal{Y}\Gamma \times Tm(A)$$

where $\mathcal{Y}$ is the yoneda embedding. We can ask for additional type structure on a simply typed category with families.

1. Product type: $Tm(A \times B) \cong Tm(A) \times Tm(B)$
2. Unit type : $Tm(\mathbf{1}) \cong \mathbf{1}$
3. Function type : $Tm(A \to B) \cong Cont\ A\ B$
4. Sum type : $Cont\ (A + B)\ C \cong Cont\ A\ C\ \times\ Cont\ B\ C$
5. Empty type : $Cont\ \mathbf{0}\ C \cong \mathbf{1}$

Note that $\cong$ here is natural isomophism of presheaves, $Cont\ A\ B := Tm(\_ \times A, B)$ and $Tm(\Gamma, A)$ is notation for $(Tm(A))_0(\Gamma)$ or the action on objects of the presheaf $Tm(A)$ on an object $\Gamma : Ob\ S_c$. Additionally, we can think of terms $Tm(\Gamma, A)$ as squiggly arrows $\Gamma \rightsquigarrow A$.

2 Syntactic Model

Simply typed lambda calculus, generated by any signature Σ, forms a simple category with families.

Construction 2.1. Syntactic Model

1. S_c is a category of contexts and substitutions
2. S_t is the set of types
3. $Tm(\Gamma, A) := \{\Gamma \vdash \cdot : A\}$
4. The empty context $\emptyset$ is the terminal object of S_c
5. Operation $\Gamma \times A$ is given by context extension.
6. Each of the type structures above corresponds to the appropriate lambda calculus constuction.

This isn't just any simply typed category with families, it is *the* one that is freely generated by the signature Σ. Thus, given an interpretation of the signature Σ, we can extend the interpretation to a functor of scwfs where the domain of the functor is the freely generated scwf. The remainder of the lecture is focused on demonstrating this fact. Specifically, given $\sigma : \text{Interp}(\Sigma)$ in S, we define $[\![\_]\!]_\sigma : STLC(\Sigma) \to S$ and show its essential uniqueness.

3 Maps out of the Free Model

Given S : scwf that has whatever type structure our lambda calculus has, we want to define a *Semantics Functor* out of the free model.

3.1 Signature Interpretation

Fist we need to know what it means to have an interpretation of a signature Σ in S.

Definition 3.1. A type interpretation of Σ in S is an assignment $\sigma_0 : \Sigma_0 \to S_t$.

Given a type interpretation $\sigma_0 : \Sigma_0 \to S_t$ we can define:

$[\![\_]\!]_t : STLC(\Sigma)_t \to S_t$ an interpretation for all types in STLC(Σ)
$[\![\_]\!]_c : STLC(\Sigma)_c \to (S_c)_0$ an interpretation of contexts in STLC(Σ)

Definition 3.2. *full* type interpretation:

$$
\begin{aligned}
&[\![X]\!]_t := \sigma_0(X) \\
&[\![\mathbf{1}]\!]_t := \mathbf{1}_s \\
&[\![A \times B]\!]_t := [\![A]\!]_t \times [\![B]\!]_t \\
&[\![A \to B]\!]_t := [\![A]\!]_t \to [\![B]\!]_t \\
&\cdots
\end{aligned}
$$

Definition 3.3. Context interpretation:

$$[\![\emptyset]\!]_c := \bullet_s \text{ terminal object in } S_c$$
$$[\![\Gamma, x : A]\!]_c := [\![\Gamma]\!]_c \times_s [\![A]\!]_t$$

Remark 3.1. This is *the* unique definition that will preserve all the structure. To be demonstrated later. We can extend the type interpretation to a function symbol interpretation.

Definition 3.4. Function symbol interpretation.

$$\forall (f : A_1, A_2, \cdots \Rightarrow B) : \Sigma_1. \quad \sigma_1(f) : Tm_s([\![A_1, A_2, \cdots]\!]_c, [\![B]\!]_t)$$

Next, we define the term translation $[\![\_]\!]_{tm} : \Gamma \vdash M : A \to [\![M]\!] : Tm_s([\![\Gamma]\!]_c, [\![A]\!]_t)$ and substitution translation $[\![\_]\!]_{sub} : (\gamma : \Delta \to \Gamma) \to [\![\gamma]\!] : S_c[[\![\Delta]\!]_c, [\![\Gamma]\!]_c]$.

3.1.1 Term Translation

Variables

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A}$$

We need to translate $\Gamma \vdash x : A$ to a term $Tm_s([\![\Gamma]\!]_c, [\![A]\!]_t)$ in S. To do so, we induct on the hypothesis $x : A \in \Gamma$ which can be broken into two cases:

$$\frac{}{x : A \in \Gamma, x : A}\ \text{HERE} \qquad\qquad \frac{x : A \in \Gamma}{x : A \in \Gamma, y : B}\ \text{LATER}$$

In the first case, we have $\Gamma, x : A \vdash x : A$ which we can interpret as the second projection of $\times_s$. For the second case, we have as a hypothesis $[\![x]\!]^{\Gamma}_{tm} : Tm_s([\![\Gamma]\!]_c, [\![A]\!]_t)$ and we need to construct $[\![x]\!]^{\Gamma, y:B}_{tm} : Tm_s([\![\Gamma, y : B]\!]_c, [\![A]\!]_t)$. Use the hypotheses and the first projection of $\times_s$

$$[\![\Gamma]\!]_c \times_s [\![B]\!]_t \xrightarrow{\pi_1} [\![\Gamma]\!]_c \rightsquigarrow [\![A]\!]_t$$

where $[\![x]\!]^{\Gamma}_{tm} : [\![\Gamma]\!]_c \rightsquigarrow [\![A]\!]_t$.

Function Symbol

$$\frac{\Gamma \vdash M_i : A_i \qquad f : A_1, \cdots \Rightarrow B}{\Gamma \vdash f(M_i, \cdots) : B}$$

We have the subterms M_i by induction and we have the interpretation of $f \in \Sigma_1$.

$$\frac{[\![M_i]\!] : [\![\Gamma]\!]_c \to [\![A_i]\!]_t \qquad \sigma_1(f) : Tm_s(\bullet_s \times_s [\![A_1]\!]_t \times_s \cdots, [\![B]\!]_t)}{?? : Tm([\![\Gamma]\!]_c, [\![B]\!]_t)}$$

We can define a susbstitution,

$$(!, [\![M_1]\!]_{tm}, [\![M_2]\!]_{tm}, \cdots) : [\![\Gamma]\!]_c \to \bullet_s \times_s [\![A_1]\!]_t \times_s \cdots$$

and act on this susbstitution with $(\sigma_1(f))_1$ to get $\sigma_1(f) \circ (!, [\![M_1]\!]_{tm}, [\![M_2]\!]_{tm}, \cdots) : Tm([\![\Gamma]\!]_c, [\![B]\!]_t)$.

Products

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash (M, N) : A \times B}$$

we have $[\![M]\!]_{tm} : Tm_s([\![\Gamma]\!]_c, [\![A]\!]_t)$ and $[\![N]\!]_{tm} : Tm_s([\![\Gamma]\!]_c, [\![B]\!]_t)$ by induction and we need to produce a term $?? : Tm_s([\![\Gamma]\!]_c, [\![A \times B]\!]_t) = Tm_s([\![\Gamma]\!]_c, [\![A]\!]_t \times [\![B]\!]_t)$. Given the fact that S has a product structure, we have the natural isomorphism

$$Tm_s([\![A]\!]_t \times [\![B]\!]_t) \cong Tm_s([\![A]\!]_t) \times Tm_s([\![B]\!]_t)$$

In particular we have a map

$$(\_, \_) : Tm_s([\![A]\!]_t) \times Tm_s([\![B]\!]_t) \to Tm_s([\![A]\!]_t \times [\![B]\!]_t)$$

which is the *backwards* direction of this isomorphism. This map is natural transformation, so we can apply it at $[\![\Gamma]\!]_c$ to get

$$(\_, \_)([\![\Gamma]\!]_c) : Tm_s([\![\Gamma]\!]_c, [\![A]\!]_t) \times Tm_s([\![\Gamma]\!]_c, [\![B]\!]_t) \to Tm_s([\![\Gamma]\!]_c, [\![A]\!]_t \times [\![B]\!]_t)$$

therefore we use $([\![M]\!]_{tm}, [\![N]\!]_{tm})([\![\Gamma]\!]_C) : Tm_s([\![\Gamma]\!]_c, [\![A]\!]_t \times [\![B]\!]_t)$

Functions

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

We have the hypothesis $[\![M]\!]_{tm} : Tm_s([\![\Gamma]\!]_c \times_s [\![A]\!]_t, [\![B]\!]_t)$ and need to construct a term $?? : Tm_s([\![\Gamma]\!]_c, [\![A]\!]_t \to [\![B]\!]_t)$. Again, we look at the natural isomorphism we have for the function type structure.

$$\lambda^{-1} : Tm_s([\![A]\!]_t \to [\![B]\!]_t) \cong Cont\ [\![A]\!]_t\ [\![B]\!]_t\ : \lambda$$

So we can use the natural transformation λ at $[\![\Gamma]\!]_c$

$$\lambda([\![\Gamma]\!]_c) : Tm_s([\![\Gamma]\!]_c \times_s [\![A]\!]_t, [\![B]\!]_t) \to Tm_s([\![\Gamma]\!]_c, [\![A]\!]_t \to [\![B]\!]_t)$$

applied to $[\![M]\!]_{tm}$. For application

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

we have $[\![M]\!]_{tm} : Tm_s([\![\Gamma]\!]_c, [\![A]\!]_t \to [\![B]\!]_t)$ and $[\![N]\!]_{tm} : Tm_s([\![\Gamma]\!]_c, [\![A]\!]_t)$. We have

$$\lambda^{-1}([\![M]\!]_{tm}) : Tm_s([\![\Gamma]\!]_c \times_s [\![A]\!]_t, [\![B]\!]_t)$$

and we can define a substitution

$$(id, [\![N]\!]_{tm}) : [\![\Gamma]\!]_c \to [\![\Gamma]\!]_c \times_s [\![A]\!]_t$$

so we can use the functorial action of $\lambda^{-1}([\![M]\!]_{tm})$, a.k.a, semantic substitutiton.

$$\lambda^{-1}([\![M]\!]_{tm}) \circ (id, [\![N]\!]_{tm}) : Tm_s([\![\Gamma]\!]_c, [\![B]\!]_t)$$

3.1.2 Substitution Translation

Given a syntactic substitution $\gamma : \Delta \to \Gamma$, we want to define a semantic substitution $[\![\gamma]\!] : [\![\Delta]\!]_c \to [\![\Gamma]\!]_c$. We define this by induction on Γ. Consider the case $\Gamma = \emptyset$. Then $[\![\gamma]\!] : [\![\Delta]\!]_c \xrightarrow{!} \bullet_s$ is *the* unique map into the terminal object $\bullet_s$. Consider the case $\Gamma = \Gamma, x : A$. The syntactic substitution is then of the form $(\delta, M/x) : \Delta \to \Gamma, x : A$ giving us $[\![\delta]\!]_{sub} : [\![\Delta]\!]_c \to [\![\Gamma]\!]_c$ and $[\![M]\!] : Tm_s([\![\Delta]\!]_c, [\![A]\!]_t)$. Using the universal property of $\times_s$, we have:

$$([\![\delta]\!], [\![M]\!]) : [\![\Delta]\!]_c \to [\![\Gamma]\!]_c \times_s [\![A]\!]_t$$

Remark 3.2. If S is the democratic scwf of sets and functions, then the model we have described here is exactly the set theoretic model of lambda calculus we have defined in previous lectures.

4 Context Interpretation is a Functor

Theorem 4.1. Here we sketch the proof that $[\![\_]\!]_c : \mathrm{STLC}(\emptyset)_c \to S_c$ is a functor.

Proof. We define functor $[\![\_]\!]_c$ as:

$$([\![\_]\!]_c)_0 := [\![\_]\!]_c \text{ the function on contexts we defined above}$$
$$([\![\_]\!]_c)_1 := [\![\_]\!]_{sub}$$

We need to show preservation of identity $[\![id_\Gamma]\!]_c = id_{[\![\Gamma]\!]}$ and preservation of composition $[\![\gamma \circ \delta]\!]_c = [\![\gamma]\!]_c \circ [\![\delta]\!]_c$. First consider preservation of identity by induction on Γ.

1. *Case* $\Gamma = \emptyset$: In this case, we are done since $[\![\Gamma]\!] = \bullet_s$ so $[\![id_\Gamma]\!]_c : \bullet_s \to \bullet_s$ which must be the unique map into the terminal object.

2. *Case* $\Gamma = \Gamma, x : A$: We have $id_\Gamma = (id_\Gamma, x/x)$.. but not quite! We actually have $id_\Gamma = (wkid_\Gamma, x/x)$ where $wkid_\Gamma : \Gamma \times A \to \Gamma$. So we need some lemmas about weakening of substitution.

Recall the weakening rule:

$$\frac{\Gamma \vdash M : A}{\Gamma, \Delta \vdash M : A}$$

we can define a version of this for substitutions

$$\frac{\gamma : \Delta \to \Gamma}{\gamma : \Theta, \Delta \to \Gamma}$$

What we'll want to show is that

$$[\![\gamma]\!]^{\Theta,\Delta} = [\![\gamma]\!]^{\Delta} \circ \Pi^{\Theta,\Delta}$$

a weakened semantic substitution is semantic substitution precomposed with a series of semantic projections. With that result in hand, we can demonstrate $[\![\_]\!]_c$ preserved identity. For preservation of composition, we proceed by cases on the output context. In the case it is empty, preservation of composition holds easily. The non-empty case requires some work.

$$[\![(\gamma, M/x) \circ \delta]\!]_c = [\![(\gamma, M/x)]\!]_c \circ [\![\delta]\!]_c$$

We can simplify the lhs:

$$\begin{aligned}[\![(\gamma, M/x) \circ \delta]\!]_c &= [\![(\gamma \circ \delta, M[\delta]/x)]\!] \\ &= ([\![\gamma \circ \delta]\!], [\![M[\delta]]\!])\end{aligned}$$

and the rhs:

$$\begin{aligned}[\![\gamma, M/x]\!] \circ [\![\delta]\!] &= ([\![\gamma]\!], [\![M]\!]) \circ [\![\delta]\!] \\ &= ([\![\gamma]\!] \circ [\![\delta]\!], [\![M]\!][[\![\delta]\!]])\end{aligned}$$

□

Thus reducing our goal to

$$([\![\gamma \circ \delta]\!], [\![M[\delta]]\!]) = ([\![\gamma]\!] \circ [\![\delta]\!], [\![M]\!][[\![\delta]\!]])$$

This is a pair so we can prove equality component wise. The left components are equal by our induction hypothesis. To demonstrate the equality of the right components, we need another fact about substitution.

$$\forall \gamma.(\ [\![M[\gamma]]\!] = [\![M]\!][[\![\gamma]\!]])$$

Or

$$\begin{array}{ccc} \Gamma \vdash_{\text{STLC}} M : A & \xrightarrow{[\![\text{-}]\!]} & [\![\Gamma]\!] \vdash_S [\![M]\!] : [\![A]\!] \\ {\scriptstyle Tm(A)(\gamma)}\downarrow & & \downarrow{\scriptstyle Tm([\![A]\!])([\![\gamma]\!])} \\ \Delta \vdash_{STLC} M[\gamma] : A & \xrightarrow[[\![\text{-}]\!]]{} & [\![\Delta]\!] \vdash_S [\![M[\gamma]]\!] = [\![M]\!][[\![\gamma]\!]] : [\![A]\!] \end{array}$$

This equation says that substitution commutes with the term translation, but we can formalize this in category theory as a natural transformation.

$$Tm_{STLC}(A) \Rightarrow Tm_S([\![A]\!]) \ \circ_F \ [\![\_]\!]_c^{op}$$

How does this typecheck? Well we know:

- $Tm_{\text{STLC}(A)} : \text{STLC}_c^{op} \to \mathbf{Set}$
- $Tm_S([\![A]\!]) : S_c^{op} \to \mathbf{Set}$
- $[\![\_]\!]_c : STLC_c \to S_c$

Then $[\![\_]\!]_c^{op} : STLC_c^{op} \to S_c^{op}$ and we can compose this with $Tm_s([\![A]\!])$ to get functors on the same category. So we require this condition as part of the data involved in a map between simple categories with families. The proof of this is condition is another big inductive proof.

4.1 Product Case

We will just consider one case of the proof that substitution commutes with translation.

Lemma 4.1. $[\![(M_1, M_2)[\gamma]]\!] = [\![(M_1, M_2)]\!][[\![\gamma]]\!]$

Proof.

$$\begin{aligned}
[\![(M_1, M_2)[\gamma]]\!] &= [\![(M_1[\gamma], M_2[\gamma])]\!] \\
&= ([\![M_1[\gamma]]\!], [\![M_2]\!][\gamma]) \\
&\quad \text{by the inductive hypothesis, we have:} \\
&= ([\![M_1]\!][[\![\gamma]]\!], [\![M_2]\!][[\![\gamma]]\!]) \\
&\quad \text{by naturality of pairing:} \\
&= ([\![M_1]\!], [\![M_2]\!])[[\![\gamma]]\!] \\
&= [\![(M_1, M_2)]\!][[\![\gamma]]\!]
\end{aligned}$$

□

5 Preservation of Equations

Our signature may have axioms and we need to account for those in the translation. We want that:

$$\forall (M = N \in \Sigma).\ [\![M]\!] = [\![N]\!]$$

Example 5.1. Product β rule.

$$\frac{\Gamma \vdash M_1 : A \qquad \Gamma \vdash M_2 : A'}{\Gamma \vdash \pi_1(M_1, M_2) = M_1 : A}\ \beta_1$$

Thus, we want that under translation:

$$[\![\pi_1(M_1, M_2)]\!] = [\![M_1]\!]$$

A quick calculation yields:

$$\begin{aligned}
[\![\pi_1(M_1, M_2)]\!] &= \overline{\pi_1}[\![(M_1, M_2)]\!] \\
&= \overline{\pi_1}(\overline{[\![M_1]\!], [\![M_2]\!]}) \\
&= [\![M_1]\!]
\end{aligned}$$

Using the overline notation here to distinguish between syntactic projection π_1 and semantic projection $\overline{\pi_1}$.

6 Weak and Stict Preservation of Structure

Thus far, we've defined maps

- Functor, $[\![\_]\!]_c$
- Function, $[\![\_]\!]_t$
- Natural transformation, $[\![\_]\!]_{tm}$

but how do we assert that a mapping between scwf preserves the context extension structure and the type structure? For this, look at definition 1 in homework 4. In lecture, we went over a more general concept; weak vs strict preservation of universal properties.

6.1 Preservation of Terminal Objects

What does it mean to require that a functor preserve terminal objects? Consider the following possible definitions: Given categories $\mathcal{C}, \mathcal{D}$ and a functor $F : \mathcal{C} \to \mathcal{D}$,

1. If $\mathbf{1} \in \mathcal{C}_0$ is terminal, then $F(\mathbf{1})$ is terminal.
2. If $\mathcal{C}$ comes with a terminal object $\mathbf{1} \in \mathcal{C}_0$, then $F(\mathbf{1})$ is terminal.
3. If $\mathbf{1}_c, \mathbf{1}_d$ are terminal objects in their respective categories, then $F(\mathbf{1}_c) = \mathbf{1}_d$.
4. If $\mathbf{1}_c, \mathbf{1}_d$ are terminal objects in their respective categories, then $F(\mathbf{1}_c) \cong \mathbf{1}_d$.

Definition 1 states that F preserves all terminal objects in $\mathcal{C}$. Definition 2 states that F preserves a particular terminal object in $\mathcal{C}$. Definitions 3 and 4 presume that both categories have specified terminal objects. Then we either ask that F preserves the terminal object up to equality (strict), or up to isomorphism (weak). Definitions 1, 2, and 4 are equivalent when "we have all the required data". This is because terminal objects are unique up to unique isomorphism. Definition 3 is strictly stronger. It is the only one that specifies exactly the output of F on a specified terminal object.

6.2 Preservation of Products

What does it mean to require that a functor preserve products? Given categories $\mathcal{C}, \mathcal{D}$ and a functor $F : \mathcal{C} \to \mathcal{D}$, we could make the definition: $\forall A, B : \mathcal{C}_0$, for any product $(A \times B, \pi_1, \pi_2)$, $F(A \times B)$ is a product of $F(A)$ and $F(B)$. We need to say something about what the projections of the target product is. We have a choice. Either we say there exists some projections or we can say the projections are specifically given by $F(\pi_1)$ and $F(\pi_2)$. The latter case is the one we want. That is, rather than

$$F(A \times B) \cong F(A) \times F(B)$$

we want to specify the forward direction of the isomorphism to be

$$F(A \times B) \xrightarrow{(F(\pi_1), F(\pi_2))} F(A) \times F(B)$$

This is our notion of weak preservation. For strict preservation we ask for

- $F(A \times B) = F(A) \times F(B)$
- $F(\pi_1) = \pi_1$
- $F(\pi_2) = \pi_2$