Lecture 16: Canonicity II

Lecturer: Max S. New
Scribe: Alexander Bandukwala

October 22nd, 2025

1 Part 1 recap

We continue our Logical Relations proof.

$$\text{Given } \cdot \vdash M : 1 + 1, \quad \text{we want to show that } M = \sigma_0() \text{ or } M = \sigma_1().$$

So we need to strengthen our inductive hypothesis from Lecture 15. We need an inductive hypothesis that applies to every open term. So we define these sets of canonical terms.

$$\begin{aligned} C\ell(A) &:= \{M \mid \cdot \vdash M : A\} \\ C\ell(\Gamma) &:= \{\gamma \mid \gamma : \cdot \to \Gamma\} \end{aligned}$$

$$\begin{aligned} Can[A] &: C\ell(A) \to Set \\ Can[\Gamma] &: C\ell(\Gamma) \to Set \end{aligned}$$

$$\begin{aligned} Can[1](M) &:= \{\star\} \\ Can[0](M) &:= \emptyset \\ Can[A \times B](M) &:= Can[A](\pi_1 M) \times Can[B](\pi_2 M) \\ Can[A_0 + A_1](M) &:= (M = \sigma_0 N_0 \times Can[A_0](N_0)) \uplus (M = \sigma_1 N_1 \times Can[A_1](N_1)) \\ Can[A \to B](M) &:= \prod_{N \in C\ell(A)} Can[A](N) \to Can[B](MN) \\ Can[\Gamma](\gamma) &:= \prod_{x:A \in \Gamma} Can[A](\gamma(x)) \end{aligned}$$

1.1 Fundamental Property/Lemma of the logical relation

Define by recursion for all $\Gamma \vdash M : A$

$$Can[M] : \prod_{\gamma : C\ell(\Gamma)} Can[\Gamma](\gamma) \to Can[A](M[\gamma])$$

1.2 Partial canonicity proof: case rule

Consider

$$\frac{\Gamma \vdash M : A_0 + A_1 \qquad \Gamma, x_0 : A_0 \vdash P_0 : C \qquad \Gamma, x_1 : A_1 \vdash P_1 : C}{\Gamma \vdash \text{case } \underline{M}\{\sigma_0 x_0 \to \underline{P_0} \mid \sigma_1 x_1 \to \underline{P_1}\} : C}$$

Assume the inductive hypothesis for the underlined subterms.
Given $\gamma : C\ell(\Gamma)$ and $\hat{\gamma} : Can[\Gamma](\gamma)$, we must construct

$$Can[C](\text{case } M\{\sigma_0 x_0 \to P_0 \mid \sigma_1 x_1 \to P_1\}[\gamma]).$$

By the inductive hypothesis on M,

$$Can[M](\hat{\gamma}) : Can[A_0 + A_1](M[\gamma]).$$

So $Can[A_0 + A_1](M[\gamma])$ gives either

$$(M[\gamma] = \sigma_0 N_0 \times \hat{N}_0 : Can[A_0](N_0)) \quad \text{or} \quad (M[\gamma] = \sigma_1 N_1 \times \hat{N}_1 : Can[A_1](N_1)).$$

In the first case, $M[\gamma] = \sigma_0 N_0$. By β-reduction,

$$\text{case } M[\gamma]\{\sigma_0 x_0 \to P_0[\gamma] \mid \sigma_1 x_1 \to P_1[\gamma]\} = P_0[\gamma, N_0/x_0].$$

We need a canonical proof of

$$Can[\Gamma, x_0 : A_0](\gamma, N_0/x_0).$$

This is given by $\hat{\gamma}$ together with $\hat{N}_0$. Then by the inductive hypothesis for P_0,

$$Can[C](P_0[\gamma, N_0/x_0]).$$

The second case ($M[\gamma] = \sigma_1 N_1$) is symmetric. □

1.3 Lambda expressions

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

Given $\gamma : C\ell(\Gamma)$ and $\hat{\gamma} : Can[\Gamma](\gamma)$, we must show

$$Can[A \Rightarrow B]\big((\lambda x.\, M)[\gamma]\big).$$

We have

$$(\lambda x.\, M)[\gamma] = \lambda x.\, M[\gamma, x/x].$$

By the definition of canonicity for function types, we are given

$$N : C\ell(A) \quad \text{and} \quad \hat{N} : Can[A](N),$$

and must construct

$$Can[B]\big((\lambda x.\, M[\gamma, x/x])\, N\big).$$

By β-reduction,

$$(\lambda x.\, M[\gamma, x/x])\, N = M[\gamma, N/x].$$

By the inductive hypothesis, we have a canonicality constructor for M when applied to a canonical substitution:

$$Can[M](\hat{\gamma}, \hat{N}/x),$$

which gives the desired $Can[B](M[\gamma, N/x])$. □

2 Using the Initiality Theorem to avoid induction

Work in the category **SCwF** of simple categories with families.

$$\text{STLC}[\emptyset] \longrightarrow LR,$$

where LR is an SCwF that "encodes" the logical relation.

A practical way to build such a morphism is to construct LR *over* the syntactic model, i.e. equip LR with a projection $\pi_1 : LR \to \text{STLC}[\emptyset]$ that preserves all SCwF structure: every object (type/context) of LR lies over a corresponding syntactic object, and constructors (products, sums, arrows, etc.) in LR project to the corresponding syntactic constructors. Concretely,

$$\begin{array}{c} LR \\ \downarrow \\ \text{STLC}[\emptyset] \end{array}$$

The reason for this "model-with-syntax" approach is that the interpretation of a type in LR refers to the type itself.

We know that for any ScWF that interprets all of the types that we get a homomomorphism, a functor of ScWF from the syntax into our model. And this is what we call the fundamental lemma

$$\begin{array}{c} LR \\ \downarrow{\scriptstyle \pi_1} \quad \nwarrow^{FL} \\ \text{STLC}[\emptyset] \end{array}$$

We prove the initiality lemma by recursion. The initiality lemmas is packaging up that recursion in a reusable way. And the way that initiality comes into all this is that we know there's a unique functor of SCwF from $STLC[\emptyset]$ to anything else which means the $FL\ \pi_1$ composition has to be equal to the identity. And this is going to encode the property that for every type we're defining a relation over it and for every term we're defining a canoninicity transformer about it.

2.1 The logical-relation SCwF

We define an SCwF LR that encodes the logical relation over the syntactic SCwF STLC[$\emptyset$].

$$\begin{aligned}
LR_{ty} &:= (A : STLC_{ty}) \times (C\ell(A) \to \mathbf{Set}) \\
(LR_c)_o &:= (\Gamma : STLC_{c,o}) \times (C\ell(\Gamma) \to \mathbf{Set}) \\
LR_{tm}(A, \hat{A})(\Gamma, \hat{\Gamma}) &:= (\Gamma \vdash M : A) \times (\prod_{\gamma : C\ell(\Gamma)} \hat{\Gamma}(\gamma) \to \hat{A}(M[\gamma]) \\
(LR_c)((\Delta, \hat{\Delta}), (\Gamma, \hat{\Gamma})) &:= (\gamma : \Delta \to \Gamma) \times (\prod_{x : A \in \Gamma} \prod_{\delta : C\ell(\Delta)} \hat{\Delta}(\delta) \to \hat{A}(\gamma(x)[\delta])) \\
(M, \hat{M}) \circ (\gamma, \hat{\gamma}) &:= (M[\gamma], \lambda\delta : C\ell(\Delta).\lambda\hat{\delta} : \hat{\Delta}.\hat{M}(\hat{\gamma}(\hat{\delta}))\hat{A}(M[\gamma][\delta]))
\end{aligned}$$

Here we use the notation $\hat{A} := C\ell(A) \to \mathbf{Set}$ and $\hat{\Gamma} := C\ell(\Gamma) \to \mathbf{Set}$.

2.2 Closed Terms and Families

We have a functor from the syntactic category of STLC contexts to **Set** that takes each context to its set of *closed terms*:

$$\text{STLC}[\emptyset] \xrightarrow{C\ell} \mathbf{Set}.$$

More generally, for any simple category with families S, we can define a similar construction of *closed terms*. The idea is that $C\ell$ maps each type A in S to the set of its closed terms, and each context Γ to the set of closed substitutions into it:

$$C\ell(A) := Tm^S_\emptyset(A) \qquad \text{and} \qquad C\ell(\Gamma) := S(\emptyset, \Gamma).$$

Thus, we can view $C\ell$ as a functor

$$S[\emptyset] \xrightarrow{C\ell} \mathbf{Set}.$$

The other component we need is the which is another simple category with families, called **Fam** (for "families"). In this category:

$$\mathbf{Fam} = \{(A : \mathbf{Set},\ A \to \mathbf{Set})\},$$

so each object is a set equipped with a family of sets indexed by it.

Putting these pieces together, we can view the logical relation construction LR that combines the syntax and semantics: