Lecture Notes 9
Termination for STLC

Carlo Angiuli

B522: PL Foundations
March 31, 2025

So far this semester we seen how to formally define programming languages in terms of abstract binding trees, type systems, and operational semantics, and we have learned the proof technique of *progress and preservation* for showing that a language's type system and operational semantics cohere with one another.

Type safety is an important property for a language to have, but there are many other properties we might wish to prove about a language, such as:

- that all programs terminate;
- that two particular functions or code fragments are indistinguishable;
- that it is impossible to define a function with a particular property; or
- that there are exactly n extensionally distinct functions of a certain type.

All of these properties can be established using a powerful proof technique known as the technique of *logical relations*. In today's lecture we will introduce logical relations with the goal of proving that all programs in the simply-typed λ-calculus terminate.

The history of logical relations is too lengthy to do justice to here, but some incomplete remarks are warranted. The key idea behind logical relations was invented by Tait [Tai67] in order to analyze term equivalence in System T. (The proof technique is sometimes called *Tait's method*.) Several years later, Girard generalized Tait's method by introducing *candidats de réductibilité* in his proof of strong normalization for System F [Gir71]. The term *logical relations* was introduced by Plotkin [Plo73] in his study of λ-definability, and later cemented by Statman [Sta85] who further refined the technique.

From the 1970s through the early 2000s, the technique of logical relations was used to great effect in the study of various λ-calculi, but it remained highly techncial to use logical relations to reason about languages with effects such as recursive types and general references. In 2001, Appel and McAllester [AM01] invented the technique of *step-indexed logical relations*, which can be readily adapted to support these and other realistic language features. Thanks to Appel and McAllester and subsequent researchers, especially Ahmed [Ahm06], step-indexed logical relations are currently one of our best tools for analyzing realistic programming languages.

This material does not have a direct analog in Harper [Har16], but Chapters 46–48 concern logical relations for other languages.

1 A first attempt

Before introducing the technique of logical relations, it is instructive to see what's hard about proving termination for STLC.

Remark 9.1. Recall that for our purposes, the STLC is the call-by-value programming language with $\mathtt{unit}$, $\tau_1 \to \tau_2$, and $\tau_1 \times \tau_2$, defined using a small-step operational semantics. We will take as granted the "basic" lemmas of uniqueness of types, weakening, finality of values, determinacy, and substitution; we will carefully note any appeals to canonical forms, progress, or preservation.

Definition 9.2. Given a closed term $\cdot \vdash e\ \mathsf{tm}$, we say that $e \Downarrow v$, or e *evaluates to* v, if $e \longmapsto^* v$ and $v\ \mathsf{val}$. When such a v exists, we say that e *terminates*; we may write $e \Downarrow$ if the identity of v is not important. (By determinacy and finality of values, v is unique if it exists.)

Remark 9.3. We've also used the $\Downarrow$ notation to refer to an inductively-defined evaluation judgment as an alternative to small-step operational semantics. For the purposes of this lecture we are taking the small-step operational semantics as primitive but reclaiming the $e \Downarrow v$ notation to mean iterated small-step reduction.

Let us start by recalling what type safety gives us:

Lemma 9.4 (Type soundness)**.** *Suppose* $\cdot \vdash e : \tau$ *and* $e \Downarrow v$*. Then:*

- *If* $\tau = \mathtt{unit}$*, then* $v = ()$*.*
- *If* $\tau = \tau_1 \to \tau_2$*, then* $v = \lambda x : \tau_1.e_2$*.*
- *If* $\tau = \tau_1 \times \tau_2$*, then* $v = (v_1, v_2)$ *where* $v_1\ \mathsf{val}$ *and* $v_2\ \mathsf{val}$*.*

Proof. Recalling that $e \Downarrow v$ means that $e \longmapsto^* v$ and $v\ \mathsf{val}$, we proceed by rule induction on $e \longmapsto^* v$.

- Case $\dfrac{}{e \longmapsto^* e}$:

  In this case $\cdot \vdash e : \tau$ and e val. The result follows by canonical forms.

- Case $\dfrac{e \longmapsto e' \qquad e' \longmapsto^* v}{e \longmapsto^* v}$:

  By $\cdot \vdash e : \tau$ and $e \longmapsto e'$ and preservation, we have $\cdot \vdash e' : \tau$. The result follows by the induction hypothesis for $e' \Downarrow v$. □

In other words, type soundness tells us that the shape of the value of a well-typed, terminating term is determined by its type. Combined with termination—the fact that all well-typed terms terminate—we learn that all well-typed terms terminate with a value whose shape is determined by their type. (Recall that both of these results require a well-typed term; ill-typed terms can get stuck or diverge.)

So on the one hand, termination strengthens type soundness; on the other hand, as we will discover in the following proof attempt, we cannot prove termination without appealing to some form of type soundness.

Theorem 9.5 (Termination). *If* $\cdot \vdash e : \tau$ *then* $e \Downarrow$.

Proof attempt. By rule induction on $\cdot \vdash e : \tau$.

- The VAR case cannot occur in the empty context.

- Cases $\dfrac{}{\cdot \vdash () : \mathtt{unit}}$ unit-INTRO , $\dfrac{x : \tau_1 \vdash e_2 : \tau_2}{\cdot \vdash \lambda x : \tau_1 . e_2 : \tau_1 \to \tau_2}$ →-INTRO :

  These are values.

- Case $\dfrac{\cdot \vdash e_1 : \tau_1 \qquad \cdot \vdash e_2 : \tau_2}{\cdot \vdash (e_1, e_2) : \tau_1 \times \tau_2}$ ×-INTRO :

  We must show $(e_1, e_2) \Downarrow$, given $e_1 \Downarrow v_1$ and $e_2 \Downarrow v_2$. By the operational semantics, $(e_1, e_2) \longmapsto^* (v_1, e_2) \longmapsto^* (v_1, v_2)$ and (v_1, v_2) val.

- Cases $\dfrac{\cdot \vdash e : \tau_1 \times \tau_2}{\cdot \vdash \mathtt{fst}(e) : \tau_1}$ ×-ELIM$_1$, $\dfrac{\cdot \vdash e : \tau_1 \times \tau_2}{\cdot \vdash \mathtt{snd}(e) : \tau_2}$ ×-ELIM$_2$:

  In the first case we must show $\mathtt{fst}(e) \Downarrow$ given $e \Downarrow$. By Lemma 9.4, $e \Downarrow (v_1, v_2)$ where v_1 val and v_2 val. By the operational semantics, $\mathtt{fst}(e) \longmapsto^* \mathtt{fst}((v_1, v_2)) \longmapsto v_1$. The second case is similar.

- Case $\dfrac{\cdot \vdash f : \tau_1 \to \tau_2 \qquad \cdot \vdash e_1 : \tau_1}{\cdot \vdash f\ e_1 : \tau_2}$ →-ELIM :

We must show that $f\ e_1 \Downarrow$, using the inductive hypotheses that $f \Downarrow$ and $e_1 \Downarrow v_1$. By Lemma 9.4, $f \Downarrow \lambda x : \tau_1.e_2$. By the operational semantics, $f\ e_1 \longmapsto^* (\lambda x : \tau_1.e_2)\ e_1 \longmapsto^* (\lambda x : \tau_1.e_2)\ v_1 \longmapsto e_2[v_1/x]$, but **we don't know anything about** $e_2[v_1/x]$.

2 Strengthening the IH

Zooming out, the problem with the above proof attempt is that it simply is not true that whenever f and e_1 terminate, then $f\ e_1$ terminates; we also need to know that the function f terminates *for every input*.

Remark 9.6. In fact, a similar thing happens in the cases for `fst` and `snd` even though the proof goes through: we need to know not only that e terminates but that $\texttt{fst}(e)$ terminates. In a call-by-value language, this follows from the fact that e evaluates to a pair because pairs evaluate their arguments. But in a call-by-name language, the proof attempt also fails for `fst` and `snd`.

The core idea of logical relations is to *strengthen the inductive hypothesis*.

Idea 9.7. Instead of proving that functions terminate, we should prove a strictly stronger property: that they *terminate for every input*.

It may seem counterintuitive that proving a stronger theorem will be easier, but we have seen this before: we were able to prove the substitution lemma for arbitrary contexts but unable to prove it for contexts of length one. We must prove a stronger result, but we also get a stronger inductive hypothesis.

It makes sense to ask that terms of type $\tau_1 \to \tau_2$ terminate on every possible input, but what about terms of type $\tau_1 \times \tau_2$ or `unit`? The concept of "every possible input" only makes sense for terms of function type.

Idea 9.8. As in Lemma 9.4, the property that we prove well-typed terms satisfy must be different at every type. For terms of type $\tau_1 \to \tau_2$ we will prove that they terminate for every input, but for terms of type $\tau_1 \times \tau_2$ and `unit` we must prove a different statement.

Since our termination proof went through for everything except functions, it is tempting to think that the statement we prove for product and `unit` types should just be ordinary termination. Unfortunately, this doesn't work. What if we have a pair of functions, then take a projection, then apply it to something? If all we know is that the pair of functions terminates, then we are back to square one: p terminates, so $\texttt{fst}(p)$ terminates, but why should $\texttt{fst}(p)\ e$ terminate?

In other words, the property we prove for terms of type $(\tau_1 \to \tau_2) \times (\tau_3 \to \tau_4)$ must imply that the first projection terminates when applied to a term of type

τ_1, and that the second projection terminates when applied to a term of type τ_3. Likewise, if we have a term of type $\tau_1 \to (\tau_2 \to \tau_3)$, then not only must it terminate when applied to a term of type τ_1, but the function so obtained must itself terminate when applied to a term of type τ_2.

Idea 9.9. The property that we prove holds for all $\cdot \vdash e : \tau$ must be defined by structural recursion on τ, where the case for function types involves applying e to inputs, and the case for product types involves the projections of e.

The property in question is called *hereditary termination*.

Definition 9.10 (Hereditary termination)**.** A closed term $\cdot \vdash e$ tm is *hereditarily terminating at type* τ, or $\mathbf{HT}_\tau(e)$, when:

- If $\tau = \mathtt{unit}$, then $e \Downarrow ()$.
- If $\tau = \tau_1 \to \tau_2$, then $e \Downarrow \lambda x : \tau_1.e_2$, and for all $\cdot \vdash e_1 : \tau_1$ such that $\mathbf{HT}_{\tau_1}(e_1)$, we have $\mathbf{HT}_{\tau_2}(e_2[e_1/x])$.
- If $\tau = \tau_1 \times \tau_2$, then $e \Downarrow (v_1, v_2)$ where v_1 val, v_2 val, $\mathbf{HT}_{\tau_1}(v_1)$, and $\mathbf{HT}_{\tau_2}(v_2)$.

This property is considerably more complex than termination, but it is straightforward to see that if we can prove that all closed, well-typed terms are hereditarily terminating, then this will imply (1) termination, and (2) type soundness.

Lemma 9.11. *If $\boldsymbol{HT}_\tau(e)$ then $e \Downarrow$.*

Proof. By structural recursion on τ.

- If $\tau = \mathtt{unit}$, then $\mathbf{HT}_{\mathtt{unit}}(e)$ means exactly that $e \Downarrow ()$.
- If $\tau = \tau_1 \to \tau_2$, then $\mathbf{HT}_{\tau_1 \to \tau_2}(e)$ implies in particular that $e \Downarrow \lambda x : \tau_1.e_2$.
- If $\tau = \tau_1 \times \tau_2$, then $\mathbf{HT}_{\tau_1 \times \tau_2}(e)$ implies in particular that $e \Downarrow (v_1, v_2)$. □

Exercise 9.12. Convince yourself that if we know every $\cdot \vdash e : \tau$ satisfies $\mathbf{HT}_\tau(e)$, then Lemma 9.4 directly follows. (In fact, we can even remove the hypothesis of Lemma 9.4 that $e \Downarrow v$, because e would necessarily terminate.)

3 One more thing

We are poised to prove that all closed, well-typed terms are hereditarily terminating at their type, but the theorem still isn't general enough! Let's try proving that the $\to$-INTRO rule preserves hereditary termination.

Theorem 9.13 (Hereditary termination). *If* $\cdot \vdash e : \tau$ *then* $\boldsymbol{HT}_\tau(e)$.

Proof attempt. By rule induction on the typing judgment.

- Case $\dfrac{x : \tau_1 \vdash e_2 : \tau_2}{\cdot \vdash \lambda x : \tau_1.e_2 : \tau_1 \to \tau_2}$ $\to$-INTRO :

  We must show that $\mathbf{HT}_{\tau_1 \to \tau_2}(\lambda x : \tau_1.e_2)$, i.e. that $\lambda x : \tau_1.e_2 \Downarrow \lambda x : \tau_1.e_2'$ and for all $\cdot \vdash e_1 : \tau_1$ such that $\mathbf{HT}_{\tau_1}(e_1)$, we have $\mathbf{HT}_{\tau_2}(e_2'[e_1/x])$. By finality of values we must have $e_2 = e_2'$, so it remains only to show that for all $\cdot \vdash e_1 : \tau_1$ such that $\mathbf{HT}_{\tau_1}(e_1)$, we have $\mathbf{HT}_{\tau_2}(e_2[e_1/x])$.

  Since this statement concerns $\mathbf{HT}_{\tau_2}$ for a term involving e_2, we might imagine that this should follow from the inductive hypothesis for e_2, but **there is no inductive hypothesis because e_2 isn't closed!**

The solution is to strengthen the inductive hypothesis one final time so that hereditary termination applies not only to closed terms but also to open terms. But hereditary termination talks about evaluation, and we can't evaluate open terms—so what should the generalization be? In fact, we can read it directly off of our failed proof attempt.

Idea 9.14. Rather than proving that all closed, well-typed terms are hereditarily terminating, we will prove that if we take any well-typed term and substitute hereditarily terminating terms for all of its variables, then the result is hereditarily terminating. That is, if $x_1 : \tau_1, \ldots, x_n : \tau_n \vdash e : \tau$, then for any $\cdot \vdash e_1 : \tau_1$ satisfying $\mathbf{HT}_{\tau_1}(e_1)$, …, and $\cdot \vdash e_n : \tau_n$ satisfying $\mathbf{HT}_{\tau_n}(e_n)$, we have $\mathbf{HT}_\tau(e[e_1/x_1] \ldots [e_n/x_n])$.

This idea is not as random as it may first appear. First, it strictly generalizes hereditary termination, because closed terms have no variables to be substituted. Secondly, the property of an open term that it becomes hereditarily terminating when we substitute in a hereditarily terminating term is precisely what we need to push through the $\to$-INTRO case. Thirdly, if we are looking to describe the evaluation behavior of an open term, the only way that open terms become involved with evaluation is by substituting for their free variables.

We introduce some notations to make Idea 9.14 more manageable.

Definition 9.15. A well-typed *closing substitution*, or *environment*, for a typing context Γ associates to every $x_i : \tau_i$ in Γ a closed term $\cdot \vdash e_i : \tau_i$ of the appropriate type. We write $\gamma : \Gamma$ when γ is a closing substitution for Γ, and we can define this by structural recursion on Γ as follows:

- $\cdot : \cdot$, and

- if $\gamma : \Gamma$ and $\cdot \vdash e : \tau$, then $(\gamma, e/x) : (\Gamma, x : \tau)$.

Concretely, we can think of γ as an (unordered) list of pairs $e_1/x_1, e_2/x_2, \ldots$. When $\gamma : \Gamma$, the set of variables in Γ must be the same as the set of variables to the right of / in γ, and each term's type must match the corresponding variable in Γ.

Remark 9.16. Although closing substitutions are not terms, the notation $\gamma : \Gamma$ is chosen intentionally to overlap with the term typing judgment. We can think of the context $x_1 : \tau_1, \ldots, x_n : \tau_n$ as a single type $\tau_1 \times \cdots \times \tau_n$, and a closing substitution for that context as a closed term of that type.

Definition 9.17. Given a term $\Gamma \vdash e : \tau$ and a closing substitution $\gamma : \Gamma$ for its context, we can perform a *simultaneous substitution* $e[\gamma]$ of all the terms in γ for their corresponding variables. We can define $e[\gamma]$ by structural recursion on γ:

- $e[\cdot] = e$, and
- $e[\gamma, e'/x] = (e[\gamma])[e'/x]$.

Lemma 9.18 (Simultaneous substitution)**.** *If* $\Gamma \vdash e : \tau$ *and* $\gamma : \Gamma$*, then* $\cdot \vdash e[\gamma] : \tau$*.*

Proof: by induction on γ, using the substitution lemma.

Finally, we can define hereditary termination for a closing substitution as pointwise hereditary termination.

Definition 9.19. A closing substitution $\gamma : \Gamma$ is *hereditarily terminating at context* Γ, or $\mathbf{HT}_\Gamma(\gamma)$, as follows:

- $\mathbf{HT}_\cdot(\cdot)$, and
- $\mathbf{HT}_{\Gamma, x:\tau}(\gamma, e/x)$ if $\mathbf{HT}_\Gamma(\gamma)$ and $\mathbf{HT}_\tau(e)$.

4 The fundamental theorem

At long last we are ready to state a theorem that we will be able to prove by rule induction, namely the statement we described in Idea 9.14 but rephrased using hereditarily terminating closing substitutions. It is often known grandly as the *fundamental theorem of logical relations*.

Theorem 9.20 (Fundamental theorem of logical relations)**.** *Suppose* $\Gamma \vdash e : \tau$ *and* $\gamma : \Gamma$ *and* $\boldsymbol{HT}_\Gamma(\gamma)$*. Then* $\boldsymbol{HT}_\tau(e[\gamma])$*.*

Rather than aborting yet another proof attempt, we note at the outset that our proof will require two straightforward technical lemmas stating that hereditary termination is closed under forward and backward evaluation. These essentially follow from the fact that for every τ, $\mathbf{HT}_\tau(e)$ is defined purely in terms of $e \Downarrow v$.

Lemma 9.21 (Head expansion)**.** *If $\boldsymbol{HT}_\tau(e')$ and $e \longmapsto e'$ then $\boldsymbol{HT}_\tau(e)$.*

Proof. By cases on τ.

- If $\tau = \mathtt{unit}$, then we know $e' \Downarrow ()$ and $e \longmapsto e'$ and must show that $e \Downarrow ()$. This is immediate by the definition of $\Downarrow$.
- If $\tau = \tau_1 \to \tau_2$, then we know $e' \Downarrow \lambda x : \tau_1.e_2$ where e_2 satisfies some property. But if $e \longmapsto e'$ then $e \Downarrow \lambda x : \tau_1.e_2$ for the same e_2.
- If $\tau = \tau_1 \times \tau_2$, then we know $e' \Downarrow (v_1, v_2)$ and v_1 and v_2 satisfy some properties. But if $e \longmapsto e'$ then $e \Downarrow (v_1, v_2)$ for the same v_1 and v_2. □

Lemma 9.22 (Head reduction)**.** *If $\boldsymbol{HT}_\tau(e)$ and $e \longmapsto e'$ then $\boldsymbol{HT}_\tau(e')$.*

Proof. By cases on τ.

- If $\tau = \mathtt{unit}$, then we know $e \Downarrow ()$ and $e \longmapsto e'$, and must show $e' \Downarrow ()$. By inversion on $e \Downarrow ()$, either $e = ()$ which contradicts $e \longmapsto e'$ by finality of values, or $e \longmapsto e''$ and $e'' \longmapsto^* ()$. By determinacy, $e' = e''$, so $e' \longmapsto^* ()$ as required.
- The cases for $\tau = \tau_1 \to \tau_2$ and $\tau = \tau_1 \times \tau_2$ are analogous. □

Corollary 9.23. *If $e \longmapsto^* e'$ then $\boldsymbol{HT}_\tau(e) \iff \boldsymbol{HT}_\tau(e')$.*

Proof: induction on $\longmapsto^*$, using the above lemmas.

Corollary 9.24. *If $\boldsymbol{HT}_\tau(e)$, then $e \Downarrow v$ and $\boldsymbol{HT}_\tau(v)$.*

Proof. Suppose $\mathbf{HT}_\tau(e)$. By Lemma 9.11 we have $e \Downarrow v$, and by Corollary 9.23 we have $\mathbf{HT}_\tau(v)$. □

With all the boring lemmas out of the way, let's prove the fundamental theorem.

Proof of Theorem 9.20. By rule induction on $\Gamma \vdash e : \tau$.

- Case $\dfrac{}{\Gamma, x : \tau \vdash x : \tau}$ VAR :

  We must show that for any $\gamma : (\Gamma, x : \tau)$ such that $\mathbf{HT}_{\Gamma,x:\tau}(\gamma)$, we have $\mathbf{HT}_\tau(x[\gamma])$. By the definition of $\mathbf{HT}_{\Gamma,x:\tau}(\gamma)$, γ must be of the form $\gamma', e/x$ where $\mathbf{HT}_\tau(e)$. Since $x[\gamma', e/x] = e$, the case follows immediately from our hypothesis $\mathbf{HT}_\tau(e)$.

- Case $\dfrac{}{\Gamma \vdash () : \mathtt{unit}}$ $\mathtt{unit}$-INTRO :

  We must show that for any $\gamma : \Gamma$ such that $\mathbf{HT}_\Gamma(\gamma)$, we have $\mathbf{HT}_{\mathtt{unit}}(()[\gamma])$. Since $()$ has no variables, $()[\gamma] = ()$. Expanding the definition of $\mathbf{HT}_{\mathtt{unit}}$, we must show that $() \Downarrow ()$, which is immediate.

- Case $\dfrac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2}$ $\times$-INTRO :

  We must show that for any $\gamma : \Gamma$ such that $\mathbf{HT}_\Gamma(\gamma)$, $\mathbf{HT}_{\tau_1 \times \tau_2}((e_1, e_2)[\gamma])$. We have two inductive hypotheses, namely that for any hereditarily terminating γ' we have $\mathbf{HT}_{\tau_1}(e_1[\gamma'])$ and $\mathbf{HT}_{\tau_2}(e_2[\gamma'])$. In particular, by setting $\gamma' = \gamma$ and Corollary 9.24, we have $e_1[\gamma] \Downarrow v_1$, $\mathbf{HT}_{\tau_1}(v_1)$, $e_2[\gamma] \Downarrow v_2$, and $\mathbf{HT}_{\tau_2}(v_2)$.

  By the definition of substitution, $(e_1, e_2)[\gamma] = (e_1[\gamma], e_2[\gamma])$. Expanding the definition of $\mathbf{HT}_{\tau_1 \times \tau_2}(e_1[\gamma], e_2[\gamma])$, we must prove that $(e_1[\gamma], e_2[\gamma]) \Downarrow (v_1', v_2')$ where v_1' val, v_2' val, $\mathbf{HT}_{\tau_1}(v_1')$, and $\mathbf{HT}_{\tau_2}(v_2')$. This follows from the operational semantics and our inductive hypotheses: $(e_1[\gamma], e_2[\gamma]) \longmapsto^* (v_1, e_2[\gamma]) \longmapsto^* (v_1, v_2)$ and (v_1, v_2) val.

- Case $\dfrac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \mathtt{fst}(e) : \tau_1}$ $\times$-ELIM$_1$:

  We must show that for any $\gamma : \Gamma$ such that $\mathbf{HT}_\Gamma(\gamma)$, $\mathbf{HT}_{\tau_1}(\mathtt{fst}(e)[\gamma])$. Our IH is that for any hereditarily terminating γ', $\mathbf{HT}_{\tau_1 \times \tau_2}(e[\gamma'])$. Setting $\gamma' = \gamma$, we conclude that $e[\gamma] \Downarrow (v_1, v_2)$ where v_1 val and $\mathbf{HT}_{\tau_1}(v_1)$. The result follows by $\mathtt{fst}(e)[\gamma] = \mathtt{fst}(e[\gamma]) \longmapsto^* \mathtt{fst}((v_1, v_2)) \longmapsto v_1$, $\mathbf{HT}_{\tau_1}(v_1)$, and Corollary 9.23.

- Case $\dfrac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \mathsf{snd}(e) : \tau_2}$ $\times$-ELIM$_2$:

  Analogous to previous case.

- Case $\dfrac{\Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash \lambda x : \tau_1 . e_2 : \tau_1 \to \tau_2}$ $\to$-INTRO :

  We must show that for any $\gamma : \Gamma$ such that $\mathbf{HT}_\Gamma(\gamma)$, $\mathbf{HT}_{\tau_1 \to \tau_2}((\lambda x : \tau_1 . e_2)[\gamma])$. Our IH is that for any $\gamma' : (\Gamma, x : \tau_1)$ such that $\mathbf{HT}_{\Gamma, x:\tau_1}(\gamma')$, $\mathbf{HT}_{\tau_2}(e_2[\gamma'])$.

  By the definition of substitution, $(\lambda x : \tau_1 . e_2)[\gamma] = \lambda x : \tau_1 . e_2[\gamma]$. Expanding the definition of $\mathbf{HT}_{\tau_1 \to \tau_2}$, we must show that $\lambda x : \tau_1 . e_2[\gamma] \Downarrow \lambda x : \tau_1 . e_2[\gamma]$ (which is immediate), and that for all $\cdot \vdash e_1 : \tau_1$ such that $\mathbf{HT}_{\tau_1}(e_1)$, we have $\mathbf{HT}_{\tau_2}(e_2[\gamma][e_1/x])$. Now suppose indeed that $\cdot \vdash e_1 : \tau_1$ and $\mathbf{HT}_{\tau_1}(e_1)$.

We can set γ' in our IH to be $\gamma, e_1/x$ because $\mathbf{HT}_{\Gamma,x:\tau_1}(\gamma, e_1/x)$. Because $e_2[\gamma, e_1/x] = e_2[\gamma][e_1/x]$, the result follows immediately from the IH.

- Case $\dfrac{\Gamma \vdash f : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_1 : \tau_1}{\Gamma \vdash f\ e_1 : \tau_2}\ \to\text{-ELIM}$:

  We must show that for any $\gamma : \Gamma$ such that $\mathbf{HT}_\Gamma(\gamma)$, $\mathbf{HT}_{\tau_2}((f\ e_1)[\gamma])$. Our IHs state that for any hereditarily terminating γ', $\mathbf{HT}_{\tau_1\to\tau_2}(f[\gamma'])$ and $\mathbf{HT}_{\tau_1}(e_1[\gamma'])$. Setting $\gamma' = \gamma$, our second IH and Corollary 9.24 give us $e_1[\gamma] \Downarrow v_1$ and $\mathbf{HT}_{\tau_1}(v_1)$; our first IH gives us $f[\gamma] \Downarrow \lambda x : \tau_1.e_2$ and, plugging in v_1, $\mathbf{HT}_{\tau_2}(e_2[v_1/x])$. The result then follows from Corollary 9.23 and $(f\ e_1)[\gamma] = f[\gamma]\ e_1[\gamma] \longmapsto^* (\lambda x : \tau_1.e_2)\ e_1[\gamma] \longmapsto^* (\lambda x : \tau_1.e_2)\ v_1 \longmapsto e_2[v_1/x]$. □

Termination (Theorem 9.5) is an immediate corollary of Theorem 9.20.

Proof of Theorem 9.5. Suppose that $\cdot \vdash e : \tau$. By Theorem 9.20, $\cdot : \cdot$, and $\mathbf{HT}.(\cdot)$, we have $\mathbf{HT}_\tau(e)$. By Lemma 9.11, this implies $e \Downarrow$. □

Remark 9.25. Some authors separate each of the cases of the proof of the fundamental theorem into separate lemmas called *compatibility lemmas*.

5 What's a logical relation?

The hereditary termination predicate $\mathbf{HT}_\tau(-)$ is a prototypical example of a *unary logical relation* (or *logical predicate*). But what exactly is a unary logical relation? Like many other terms in this class (e.g., operational semantics) there is not an exact definition, but unary logical relations generally have the following characteristics:

- The logical predicate itself is a family of type-indexed predicates over closed terms, defined by structural recursion on types.
- The logical predicate implies the property we are trying to prove, but unlike the property of interest, can be proven by rule induction.
- The definition of the logical predicate at each type reflects the type structure itself (this is the *logical* part) and is designed to be closed under elimination principles, perhaps unlike the property we are trying to prove.
- The logical predicate is defined in terms of evaluation behavior and is therefore closed under head expansion (and sometimes head reduction), which is used in the proof of the fundamental theorem.

- We extend the logical predicate to open terms by quantifying over all closing substitutions that pointwise satisfy the logical predicate.

Binary logical relations satisfy the same properties, except that they are binary relations rather than unary relations (predicates). Our next example of a logical relation will be binary.

Remark 9.26. There are several ways to define hereditary termination for the STLC.

References

[Ahm06] Amal Ahmed. "Step-Indexed syntactic logical relations for recursive and quantified types". In: *Proceedings of the 15th European Conference on Programming Languages and Systems*. ESOP'06. Vienna, Austria: Springer-Verlag, 2006, pp. 69–83. ISBN: 354033095X. DOI: 10.1007/11693024_6.

[AM01] Andrew W. Appel and David McAllester. "An indexed model of recursive types for foundational proof-carrying code". In: *ACM Transactions on Programming Languages and Systems* 23.5 (Sept. 2001), pp. 657–683. ISSN: 0164-0925. DOI: 10.1145/504709.504712.

[Gir71] Jean-Yves Girard. "Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types". In: *Proceedings of the Second Scandinavian Logic Symposium*. Ed. by J.E. Fenstad. Vol. 63. Studies in Logic and the Foundations of Mathematics. Elsevier, 1971, pp. 63–92. DOI: 10.1016/S0049-237X(08)70843-7.

[Har16] Robert Harper. *Practical Foundations for Programming Languages*. Second Edition. Cambridge University Press, 2016. ISBN: 9781107150300. DOI: 10.1017/CBO9781316576892.

[Plo73] G. D. Plotkin. "Lambda Definability and Logical Relations". Unpublished manuscript. 1973. URL: https://homepages.inf.ed.ac.uk/gdp/publications/logical_relations_1973.pdf.

[Sta85] R. Statman. "Logical relations and the typed λ-calculus". In: *Information and Control* 65.2 (1985), pp. 85–97. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(85)80001-2.

[Tai67] W. W. Tait. "Intensional interpretations of functionals of finite type I". In: *Journal of Symbolic Logic* 32.2 (1967), pp. 198–212. DOI: 10.2307/2271658.