Lecture Notes 12
Parametricity

Carlo Angiuli

B522: PL Foundations
April 20, 2026

these notes are short on words...

In this lecture, we will prove, or at least sketch the proof of, the *parametricity theorem* for System F (Theorem 12.22), which states that a certain binary logical relation is reflexive. The key idea behind parametricity is that we can "instantiate" type variables in System F not only with types but with a much larger class of type-like relations. From the parametricity theorem we will deduce that System F is terminating, that open logical equivalence is sound for observational equivalence, and that a number of free theorems and representation independence results hold.

This lecture will draw heavily upon ideas from the past three lectures on termination, observational equivalence, and System F. The parametricity theorem and its consequences are covered in Chapter 48 of Harper [Har16], although several of our key definitions differ, more closely resembling Sections 4 and 5 of Lau Skorstengaard's notes from Amal Ahmed's OPLSS lectures on logical relations.

1 Termination and *candidats de réducibilité*

The goal of the parametricity theorem is to provide a sound characterization of observational equivalence in System F, so we will once again extend our language with a type `ans` with two values `yes` and `no`. As with "STLC++," this lets us easily define a notion of program outcome with two distinct possibilities.

The statement of the parametricity theorem is quite involved, so we will start by discussing the key idea in isolation. Let's imagine trying to extend our termination proof for STLC—which you may recall involved a unary logical relation called *hereditary termination*—to System F. (Our goal is not in fact to prove termination, although we will end up proving termination for `ans` *en passant*.)

Just as in the STLC, we cannot prove termination by a direct inductive argument because the conjunction of $f \Downarrow$ and $e \Downarrow$ does not imply $f\ e \Downarrow$. In order to apply the technique of hereditary termination, we need to decide what it means for a term to be hereditarily terminating at type $\forall\alpha.\tau$. A natural but flawed choice is the following:

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

- If $\tau = \mathsf{ans}$, then $e \Downarrow \mathtt{yes}$ or $e \Downarrow \mathtt{no}$.
- If $\tau = \tau_1 \to \tau_2$, then for all $\cdot \vdash e_1 : \tau_1$ such that $\mathbf{HT}_{\tau_1}(e_1)$, we have $\mathbf{HT}_{\tau_2}(e\ e_1)$.
- If $\tau = \forall\alpha.\tau_2$, then for all $\cdot \vdash \tau_1\ \mathsf{ty}$, $\mathbf{HT}_{\tau_2[\tau_1/\alpha]}(e@\tau_1)$.

The issue with this definition is subtle but fatal: it is actually circular! We are defining $\mathbf{HT}_\tau$ by structural recursion on τ, but the definition of $\mathbf{HT}_{\forall\alpha.\tau_2}$ makes reference to every $\mathbf{HT}_{\tau_2[\tau_1/\alpha]}$ which includes type expressions that are just as large or even larger than $\forall\alpha.\tau_2$ itself! Compare this to the definition of $\mathbf{HT}_{\tau_1\to\tau_2}$, which refers only to $\mathbf{HT}_{\tau_1}$ and $\mathbf{HT}_{\tau_2}$.

To see concretely where this goes wrong, let us rephrase the definition as an infinite conjunction: we define $\mathbf{HT}_{\forall\alpha.\alpha}(e)$ to hold if and only if $\mathbf{HT}_{\mathsf{ans}}(e@\mathsf{ans})$ holds, and $\mathbf{HT}_{\mathsf{ans}\to\mathsf{ans}}(e@(\mathsf{ans} \to \mathsf{ans}))$ holds, and $\mathbf{HT}_{\forall\alpha.\alpha}(e@(\forall\alpha.\alpha))$ holds, and $\mathbf{HT}_{\forall\alpha.\alpha\to\alpha}(e@(\forall\alpha.\alpha \to \alpha))$ holds, and…

Remark 12.1. The fact that $\forall\alpha.\tau$ quantifies over all types—including $\forall\alpha.\tau$—is known as *impredicative* quantification/polymorphism. The distinction between impredicative and predicative quantification is a big topic, but for now we simply note:

- Impredicativity is an essential ingredient of our Church encodings. Defining the function $\mathtt{not} : \mathtt{bool} \to \mathtt{bool}$, for example, requires instantiating the type quantifier of a term of type $\mathtt{bool} = \forall\alpha.\alpha \to \alpha \to \alpha$ at $\mathtt{bool}$ itself.
- On the other hand, the parametric nature of System F's polymorphism is quite separate from its impredicative nature; it is possible to study generics and free theorems in the setting of what is called *predicative* System F.

In much the same way that our earlier proof by logical relations upgrades termination into the much stronger condition of hereditary termination, Girard's [Gir71] famous solution to this circularity is to significantly strengthen the definition of $\mathbf{HT}_{\forall\alpha.\tau}$ by allowing α to range not over all *types* but in effect over all *possible logical relations*, his so-called *candidats de réductibilité*.

Explain: to understand this move, consider that we would like the definition of $\mathbf{HT}_{\forall\alpha.\alpha\to\alpha}(f)$ to imply that for all $\mathbf{HT}_\tau(e)$, $\mathbf{HT}_\tau(f@\tau\ e)$…

2 The parametricity theorem

Considerable machinery is required to even state the parametricity theorem (Theorem 12.22), much less prove it. In this section, we will state all the necessary definitions and lemmas, and sketch some proofs. Although it is not strictly necessary, we will cover existential types explicitly so that we can establish representation independence results without needing to unfold any Church encodings.

Definition 12.2. Given $\cdot;\cdot \vdash e : \mathsf{ans}$ and $\cdot;\cdot \vdash e' : \mathsf{ans}$ we say that e and e' are *Kleene equivalent*, written $e \simeq e'$, if $e \Downarrow \mathsf{yes}$ and $e' \Downarrow \mathsf{yes}$, or $e \Downarrow \mathsf{no}$ and $e' \Downarrow \mathsf{no}$.

Remark 12.3. It is easy to see that Kleene equivalence is symmetric, transitive, and closed under head expansion, but it is **not** at all obvious that it is reflexive, as that is tantamount to termination. We will deduce reflexivity in Corollary 12.23 as a corollary of parametricity.

Definition 12.4. A *candidate relation* or *candidate* is a triple (τ, τ', R) such that τ, τ' are closed types, R is a binary relation between closed terms of type τ and closed terms of type τ', and R is closed under head expansion in both arguments.

Definition 12.5. A *relational environment* ρ for type context Δ, written $\rho : \Delta$, assigns a candidate to each type variable in Δ. That is,

- $\cdot : \cdot$ and
- $(\rho, (\tau, \tau', R)/\alpha) : (\Delta, \alpha\ \mathsf{ty})$ if $\rho : \Delta$ and (τ, τ', R) is a candidate.

Given $\rho : \Delta$ we can extract closing type substitutions $\mathsf{lhs}(\rho), \mathsf{rhs}(\rho)$

$$\mathsf{lhs}(\cdot) = \cdot \qquad \mathsf{rhs}(\cdot) = \cdot$$
$$\mathsf{lhs}(\rho, (\tau, \tau', R)/\alpha) = \mathsf{lhs}(\rho), \tau/\alpha \qquad \mathsf{rhs}(\rho, (\tau, \tau', R)/\alpha) = \mathsf{rhs}(\rho), \tau'/\alpha$$

as well as look up a binary relation $\rho(\alpha)$ between $\alpha[\mathsf{lhs}(\rho)]$ and $\alpha[\mathsf{rhs}(\rho)]$ for any type variable α in Δ.

For $\rho : \Delta$ and $\Delta \vdash \tau\ \mathsf{ty}$ we define $e \sim e' : \tau\ [\rho]$ as a binary relation between terms $\cdot;\cdot \vdash e : \tau[\mathsf{lhs}(\rho)]$ and $\cdot;\cdot \vdash e' : \tau[\mathsf{rhs}(\rho)]$ by induction on τ.

Definition 12.6 (Closed logical equivalence)**.** We define $e \sim e' : \tau\ [\rho]$ by induction on τ as follows:

- $e \sim e' : \mathsf{ans}\ [\rho]$ when $e \simeq e'$.
- $e \sim e' : \tau_1 \to \tau_2\ [\rho]$ for all $e_1 \sim e_1' : \tau_1\ [\rho]$ we have $e\ e_1 \sim e'\ e_1' : \tau_2\ [\rho]$.

- $e \sim e' : \forall\alpha.\tau_2\ [\rho]$ when for all candidates (τ, τ', R) we have $e@\tau \sim e'@\tau' : \tau_2\ [\rho, (\tau, \tau', R)/\alpha]$.
- $e \sim e' : \exists\alpha.\tau_2\ [\rho]$ when

$$e \Downarrow \mathsf{pack}\ \langle \tau, e_2 \rangle\ \mathsf{as}\ \exists\alpha.\tau_2 \text{ and } e' \Downarrow \mathsf{pack}\ \langle \tau', e_2' \rangle\ \mathsf{as}\ \exists\alpha.\tau_2$$

  and there exists a candidate (τ, τ', R) such that $e_2 \sim e_2' : \tau_2\ [\rho, (\tau, \tau', R)/\alpha]$.
- $e \sim e' : \alpha\ [\rho]$ when $e\ \rho(\alpha)\ e'$.

Lemma 12.7 (Head expansion). *If $d \longmapsto^* e$ and $d' \longmapsto^* e'$ and $e \sim e' : \tau\ [\rho]$ then $d \sim d' : \tau\ [\rho]$.*

Corollary 12.8. *For $\rho : \Delta$ and $\Delta \vdash \tau\ \mathsf{ty}$, logical equivalence is a candidate relation $(\tau[\mathsf{lhs}(\rho)], \tau[\mathsf{rhs}(\rho)], - \sim - : \tau\ [\rho])$.*

Lemma 12.9 (Compositionality). *Suppose $\Delta, \alpha\ \mathsf{ty} \vdash \tau\ \mathsf{ty}$, $\Delta \vdash \tau_1\ \mathsf{ty}$, and $\rho : \Delta$. Then for all $\cdot;\cdot \vdash e : \tau[\tau_1/\alpha][\mathsf{lhs}(\rho)]$ and $\cdot;\cdot \vdash e' : \tau[\tau_1/\alpha][\mathsf{rhs}(\rho)]$,*

$$e \sim e' : \tau[\tau_1/\alpha]\ [\rho] \iff e \sim e' : \tau\ [\rho, (\tau_1[\mathsf{lhs}(\rho)], \tau_1[\mathsf{rhs}(\rho)], - \sim - : \tau_1\ [\rho])/\alpha]$$

Proof. By structural induction on τ; note that Corollary 12.8 permits us to extend ρ with logical equivalence at τ_1. □

For a typing context Γ well-formed in Δ and two closing substitutions γ, γ' we define logical equivalence as pointwise (heterogeneous) logical equivalence.

Definition 12.10. We define $\gamma \sim \gamma' : \Gamma\ [\rho]$ by induction on Γ as follows:

- $\cdot \sim \cdot : \cdot\ [\rho]$, and
- $(\gamma, e/x) \sim (\gamma', e'/x) : (\Gamma, x : \tau)\ [\rho]$ if $\gamma \sim \gamma' : \Gamma\ [\rho]$ and $e \sim e' : \tau\ [\rho]$.

Lemma 12.11 (Weakening). *Suppose $\rho : \Delta$ and (τ, τ', R) is a candidate. Then:*

1. *If $\Delta \vdash \tau_1\ \mathsf{ty}$ then for all $\cdot;\cdot \vdash e : \tau_1[\mathsf{lhs}(\rho)]$ and $\cdot;\cdot \vdash e' : \tau_1[\mathsf{rhs}(\rho)]$,*

$$e \sim e' : \tau_1\ [\rho] \iff e \sim e' : \tau_1\ [\rho, (\tau, \tau', R)/\alpha]$$

2. *If Γ is well-formed in Δ, then for all $\gamma : \Gamma[\mathsf{lhs}(\rho)]$ and $\gamma' : \Gamma[\mathsf{rhs}(\rho)]$,*

$$\gamma \sim \gamma' : \Gamma\ [\rho] \iff \gamma \sim \gamma' : \Gamma\ [\rho, (\tau, \tau', R)/\alpha]$$

Definition 12.12 (Open logical equivalence)**.** We say that two terms $\Delta; \Gamma \vdash e : \tau$ and $\Delta; \Gamma \vdash e' : \tau$ are *(open) logically equivalent*, written $\Delta; \Gamma \vdash e \sim e' : \tau$, if for all relational environments $\rho : \Delta$ and all $\gamma \sim \gamma' : \Gamma\ [\rho]$ we have $e[\mathsf{lhs}(\rho)][\gamma] \sim e'[\mathsf{rhs}(\rho)][\gamma'] : \tau\ [\rho]$.

We can now prove what are called *compatibility lemmas* stating that each typing rule respects open logical equivalence.

Lemma 12.13 (Compatibility for ans-$\textsc{intro}_1$)**.** $\Delta; \Gamma \vdash \mathtt{yes} \sim \mathtt{yes} : \mathsf{ans}$.

Lemma 12.14 (Compatibility for ans-$\textsc{intro}_2$)**.** $\Delta; \Gamma \vdash \mathtt{no} \sim \mathtt{no} : \mathsf{ans}$.

Lemma 12.15 (Compatibility for $\textsc{var}$)**.** $\Delta; \Gamma, x : \tau \vdash x \sim x : \tau$.

Lemma 12.16 (Compatibility for $\to$-$\textsc{intro}$)**.** *If* $\Delta; \Gamma, x : \tau_1 \vdash e_2 \sim e_2' : \tau_2$ *then* $\Delta; \Gamma \vdash \lambda x : \tau_1.e_2 \sim \lambda x : \tau_1.e_2' : \tau_1 \to \tau_2$.

Lemma 12.17 (Compatibility for $\to$-$\textsc{elim}$)**.** *If* $\Delta; \Gamma \vdash f \sim f' : \tau_1 \to \tau_2$ *and* $\Delta; \Gamma \vdash e_1 \sim e_1' : \tau_1$ *then* $\Delta; \Gamma \vdash f\ e_1 \sim f'\ e_1' : \tau_2$.

Lemma 12.18 (Compatibility for $\forall$-$\textsc{intro}$)**.** *If* $\Delta, \alpha\ \mathsf{ty}; \Gamma \vdash e \sim e' : \tau_2$ *then* $\Delta; \Gamma \vdash \Lambda\alpha.e \sim \Lambda\alpha.e' : \forall\alpha.\tau_2$.

Proof. Uses Lemma 12.11. □

Lemma 12.19 (Compatibility for $\forall$-$\textsc{elim}$)**.** *If* $\Delta; \Gamma \vdash e \sim e' : \forall\alpha.\tau$ *and* $\Delta \vdash \tau_1\ \mathsf{ty}$ *then* $\Delta; \Gamma \vdash e@\tau_1 \sim e'@\tau_1 : \tau[\tau_1/\alpha]$.

Proof. Uses Lemma 12.9. □

Lemma 12.20 (Compatibility for $\exists$-$\textsc{intro}$)**.** *If* $\Delta \vdash \tau_r\ \mathsf{ty}$, $\Delta, \alpha\ \mathsf{ty} \vdash \tau_i\ \mathsf{ty}$, *and* $\Delta; \Gamma \vdash e \sim e' : \tau_i[\tau_r/\alpha]$, *then* $\Delta; \Gamma \vdash \mathsf{pack}\ \langle \tau_r, e \rangle\ \mathsf{as}\ \exists\alpha.\tau_i \sim \mathsf{pack}\ \langle \tau_r, e' \rangle\ \mathsf{as}\ \exists\alpha.\tau_i : \exists\alpha.\tau_i$.

Proof. Uses Lemma 12.9. □

Lemma 12.21 (Compatibility for $\exists$-$\textsc{elim}$)**.** *If* $\Delta; \Gamma \vdash e \sim e' : \exists\alpha.\tau_i$, $\Delta \vdash \tau_2\ \mathsf{ty}$, *and* $\Delta, \alpha\ \mathsf{ty}; \Gamma, x : \tau_i \vdash e_2 \sim e_2' : \tau_2$, *then* $\Delta; \Gamma \vdash \mathsf{unpack}\ \langle \alpha, x \rangle = e\ \mathtt{in}\ e_2 \sim \mathsf{unpack}\ \langle \alpha, x \rangle = e'\ \mathtt{in}\ e_2' : \tau_2$.

Proof. Uses Lemma 12.11. □

Theorem 12.22 (Parametricity)**.** *If* $\Delta; \Gamma \vdash e : \tau$ *then* $\Delta; \Gamma \vdash e \sim e : \tau$.

Proof. By rule induction on the typing judgment, straightforwardly applying compatibility lemmas. □

The remainder of this lecture concerns corollaries of the parametricity theorem, starting with termination for System F.

Corollary 12.23 (Reflexivity of Kleene equivalence)**.** *If* $\cdot;\cdot \vdash e : \mathsf{ans}$ *then* $e \simeq e$.

Proof. Suppose $\cdot;\cdot \vdash e : \mathsf{ans}$. By Theorem 12.22 we have $\cdot;\cdot \vdash e \sim e : \mathsf{ans}$; unfolding definitions, this implies $e \simeq e$. □

Corollary 12.24 (Termination for ans)**.** *If* $\cdot;\cdot \vdash e : \mathsf{ans}$ *then* $e \Downarrow \mathsf{yes}$ *or* $e \Downarrow \mathsf{no}$.

3 Characterizing observational equivalence

The definition of observational equivalence for System F is essentially identical to that of STLC++, accounting only for the differences in the syntax and type system.

Definition 12.25. *Term contexts* C are defined inductively as follows:

$$\begin{array}{rrl} \textit{Term contexts} & C ::= & \circ \mid \lambda x : \tau.C \mid C\ e \mid e\ C \mid \Lambda\alpha.C \mid C@\tau \\ & \mid & \mathsf{pack}\ \langle \tau, C\rangle\ \mathsf{as}\ \exists\alpha.\tau' \mid \mathsf{unpack}\ \langle \alpha, x\rangle = C\ \mathtt{in}\ e \\ & \mid & \mathsf{unpack}\ \langle \alpha, x\rangle = e\ \mathtt{in}\ C \end{array}$$

For any term context C and term e, we define the instantiation of C with e, written $C\{e\}$, by structural recursion in the obvious way.

Definition 12.26. The judgment $C : (\Delta;\Gamma \rhd \tau) \rightsquigarrow (\Delta';\Gamma' \rhd \tau')$ is defined inductively as follows:

$$\frac{}{\circ : (\Delta;\Gamma \rhd \tau) \rightsquigarrow (\Delta;\Gamma \rhd \tau)} \qquad \frac{C : (\Delta;\Gamma \rhd \tau) \rightsquigarrow (\Delta';\Gamma', x : \tau_1 \rhd \tau_2)}{\lambda x : \tau_1.C : (\Delta;\Gamma \rhd \tau) \rightsquigarrow (\Delta';\Gamma' \rhd \tau_1 \to \tau_2)}$$

$$\frac{C : (\Delta;\Gamma \rhd \tau) \rightsquigarrow (\Delta';\Gamma' \rhd \tau_1 \to \tau_2) \qquad \Delta';\Gamma' \vdash e_1 : \tau_1}{C\ e_1 : (\Delta;\Gamma \rhd \tau) \rightsquigarrow (\Delta';\Gamma' \rhd \tau_2)}$$

$$\frac{C : (\Delta;\Gamma \rhd \tau) \rightsquigarrow (\Delta';\Gamma' \rhd \tau_1) \qquad \Delta';\Gamma' \vdash f : \tau_1 \to \tau_2}{f\ C : (\Delta;\Gamma \rhd \tau) \rightsquigarrow (\Delta';\Gamma' \rhd \tau_2)}$$

$$\frac{C : (\Delta;\Gamma \rhd \tau) \rightsquigarrow (\Delta', \alpha\ \mathsf{ty};\Gamma' \rhd \tau')}{\Lambda\alpha.C : (\Delta;\Gamma \rhd \tau) \rightsquigarrow (\Delta';\Gamma' \rhd \forall\alpha.\tau')}$$

$$\frac{C : (\Delta;\Gamma \rhd \tau) \rightsquigarrow (\Delta';\Gamma' \rhd \forall\alpha.\tau_2) \qquad \Delta' \vdash \tau_1\ \mathsf{ty}}{C@\tau_1 : (\Delta;\Gamma \rhd \tau) \rightsquigarrow (\Delta';\Gamma' \rhd \tau_2[\tau_1/\alpha])}$$

$$\frac{\Delta' \vdash \tau_r \text{ ty} \qquad \Delta', \alpha \text{ ty} \vdash \tau_i \text{ ty} \qquad C : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\Delta'; \Gamma' \rhd \tau_i[\tau_r/\alpha])}{\mathsf{pack}\ \langle \tau_r, C \rangle \ \mathsf{as}\ \exists \alpha.\tau_i : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\Delta'; \Gamma' \rhd \exists \alpha.\tau_i)}$$

$$\frac{C : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\Delta'; \Gamma' \rhd \exists \alpha.\tau_i) \qquad \Delta' \vdash \tau' \text{ ty} \qquad \Delta', \alpha \text{ ty}; \Gamma', x : \tau_i \vdash e : \tau'}{\mathsf{unpack}\ \langle \alpha, x \rangle = C\ \mathtt{in}\ e : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\Delta'; \Gamma' \rhd \tau')}$$

$$\frac{\Delta'; \Gamma' \vdash e : \exists \alpha.\tau_i \qquad \Delta' \vdash \tau' \text{ ty} \qquad C : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\Delta', \alpha \text{ ty}; \Gamma', x : \tau_i \rhd \tau')}{\mathsf{unpack}\ \langle \alpha, x \rangle = e\ \mathtt{in}\ C : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\Delta'; \Gamma' \rhd \tau')}$$

Lemma 12.27. *If* $C : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\Delta'; \Gamma' \rhd \tau')$ *and* $\Delta; \Gamma \vdash e : \tau$*, then* $\Delta'; \Gamma' \vdash C\{e\} : \tau'$*.*

Definition 12.28 (Observational equivalence)**.** Given two terms $\Delta; \Gamma \vdash e : \tau$ and $\Delta; \Gamma \vdash e' : \tau$, we say that e and e' are *observationally equivalent*, $\Delta; \Gamma \vdash e \cong e' : \tau$, if for every $C : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\cdot; \cdot \rhd \mathsf{ans})$ we have $C\{e\} \simeq C\{e'\}$.

Observational equivalence is clearly reflexive, symmetric, and transitive because Kleene equivalence is. For the same reasons as in STLC++, it is also a congruence and—crucially for soundness—the coarsest consistent congruence.

IMPORTANT: in this lecture, a "congruence" need not be reflexive, symmetric, or transitive; this is a break from the terminology from previous lectures. (I will update those in a future year.)

Lemma 12.29. *Observational equivalence is a* congruence*: if* $\Delta; \Gamma \vdash e \cong e' : \tau$ *and* $C : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\Delta'; \Gamma' \rhd \tau')$*, then* $\Delta'; \Gamma' \vdash C\{e\} \cong C\{e'\} : \tau'$*.*

Lemma 12.30. *Observational equivalence is* consistent *with Kleene equivalence: if* $\cdot; \cdot \vdash e \cong e' : \mathsf{ans}$ *then* $e \simeq e'$*.*

Lemma 12.31. *Observational equivalence is the* coarsest *consistent congruence: for any consistent congruence* R*, if* $\Delta; \Gamma \vdash e\ R\ e' : \tau$ *then* $\Delta; \Gamma \vdash e \cong e' : \tau$*.*

To prove that open logical equivalence implies observational equivalence, it thus suffices to show that open logical equivalence is a consistent congruence. Consistency follows trivially from the definition of logical equivalence, and congruence follows from the results of the previous section.

Lemma 12.32. *Open logical equivalence is a congruence: if* $\Delta; \Gamma \vdash e \sim e' : \tau$ *and* $C : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\Delta'; \Gamma' \rhd \tau')$*, then* $\Delta'; \Gamma' \vdash C\{e\} \sim C\{e'\} : \tau'$*.*

Proof. By rule induction on $C : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\Delta'; \Gamma' \rhd \tau')$, using Theorem 12.22 and the relevant compatibility lemma for each case. We illustrate a typical case:

- Case $\dfrac{C : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\Delta'; \Gamma' \rhd \tau_1 \to \tau_2) \qquad \Delta'; \Gamma' \vdash e_1 : \tau_1}{C\ e_1 : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\Delta'; \Gamma' \rhd \tau_2)}$:

  Suppose $\Delta; \Gamma \vdash e \sim e' : \tau$; we must show that $\Delta'; \Gamma' \vdash C\{e\}\ e_1 \sim C\{e'\}\ e_1 : \tau_2$. We have $\Delta'; \Gamma' \vdash C\{e\} \sim C\{e'\} : \tau_1 \to \tau_2$ by our inductive hypothesis, and $\Delta'; \Gamma' \vdash e_1 \sim e_1 : \tau_1$ by Theorem 12.22; the result follows by the compatibility lemma for $\to$-ELIM (Lemma 12.17). □

Remark 12.33. We have now shown that open logical equivalence is reflexive and a congruence. It is not hard to show that it is symmetric, but it is actually quite hard to show that it is transitive. Luckily, we do not need transitivity to establish soundness.

Theorem 12.34 (Soundness)**.** *If* $\Delta; \Gamma \vdash e \sim e' : \tau$ *then* $\Delta; \Gamma \vdash e \cong e' : \tau$.

Proof. Immediate by Lemmas 12.31 and 12.32. More explicitly, suppose that $\Delta; \Gamma \vdash e \sim e' : \tau$ and $C : (\Delta; \Gamma \rhd \tau) \rightsquigarrow (\cdot; \cdot \rhd \mathsf{ans})$; we must show that $C\{e\} \simeq C\{e'\}$. By Lemma 12.32 we have $\cdot; \cdot \vdash C\{e\} \sim C\{e'\} : \mathsf{ans}$. Unfolding definitions, this means precisely that $C\{e\} \simeq C\{e'\}$. □

4 Deriving free theorems

Famously, parametricity implies that System F's polymorphism is so strong that we can instantiate terms of type $\forall\alpha.\tau$ at arbitrary candidate relations. Free theorems generally follow directly as long as we can choose the right candidate.

Theorem 12.35 (Polymorphic identity)**.** *If* $\cdot; \cdot \vdash f : \forall\alpha.\alpha \to \alpha$*, then for all* $\cdot; \cdot \vdash e : \tau$ *we have* $f@\tau\ e \longmapsto^* e$.

Proof. By Theorem 12.22 we have $\cdot; \cdot \vdash f \sim f : \forall\alpha.\alpha \to \alpha$. Unfolding definitions, this means that for all relational environments $\cdot : \cdot$ and all closing substitutions $\cdot \sim \cdot : \cdot\ [\cdot]$ we have $f[\mathsf{lhs}(\cdot)][\cdot] \sim f[\mathsf{rhs}(\cdot)][\cdot] : \forall\alpha.\alpha \to \alpha\ [\cdot]$, i.e., we have the closed logical equivalence $f \sim f : \forall\alpha.\alpha \to \alpha\ [\cdot]$. Expanding this definition, we know that for all candidates (τ, τ', R) we have $f@\tau \sim f@\tau' : \alpha \to \alpha\ [(\tau, \tau', R)/\alpha]$.

We will choose the candidate (τ, τ, R) defined by

$$d\ R\ d' \iff (d \longmapsto^* e \text{ and } d' \longmapsto^* e)$$

It is easy to see that R is closed under head expansion. Expanding the definition of $f@\tau \sim f@\tau : \alpha \to \alpha\ [(\tau, \tau, R)/\alpha]$, we know that for any $e_1 \sim e_1' : \alpha\ [(\tau, \tau, R)/\alpha]$ we have $f@\tau\ e_1 \sim f@\tau\ e_1' : \alpha\ [(\tau, \tau, R)/\alpha]$, or rewriting once more, that for any $e_1\ R\ e_1'$ we have $(f@\tau\ e_1)\ R\ (f@\tau\ e_1')$.

We choose $e_1 = e_1' = e$ (which satisfies $e\ R\ e$), concluding $(f@\tau\ e)\ R\ (f@\tau\ e)$ and hence $f@\tau\ e \longmapsto^* e$ as required. □

So far we have used the parametricity theorem to deduce a free theorem stated in terms of evaluation behavior. By combining this result with soundness, we can deduce an even stronger version of the same free theorem stated in terms of observational equivalence.

Corollary 12.36. *If* $\cdot;\cdot \vdash f : \forall\alpha.\alpha \to \alpha$*, then* f *is observationally equivalent to the polymorphic identity function, i.e.,* $\cdot;\cdot \vdash f \cong (\Lambda\alpha.\lambda x : \alpha.x) : \forall\alpha.\alpha \to \alpha$*.*

Proof. By Theorem 12.34 it suffices to show $\cdot;\cdot \vdash f \sim (\Lambda\alpha.\lambda x : \alpha.x) : \forall\alpha.\alpha \to \alpha$. Unfolding definitions, we must show $f \sim (\Lambda\alpha.\lambda x : \alpha.x) : \forall\alpha.\alpha \to \alpha\ [\cdot]$, and hence that for all candidates (τ, τ', R) and all $e \sim e' : \alpha\ [(\tau, \tau', R)/\alpha]$, we have $f@\tau\ e \sim (\Lambda\alpha.\lambda x : \alpha.x)@\tau'\ e' : \alpha\ [(\tau, \tau', R)/\alpha]$. By head expansion on both sides, using Theorem 12.35 on the left, it suffices to show that $e \sim e' : \alpha\ [(\tau, \tau', R)/\alpha]$, which is immediate. □

Theorem 12.37 (Polymorphic equality)**.** *If* $\cdot;\cdot \vdash f : \forall\alpha.\alpha \to \alpha \to \mathsf{ans}$*, then for all* $\cdot;\cdot \vdash e_1 : \tau$*,* $\cdot;\cdot \vdash e_2 : \tau$*,* $\cdot;\cdot \vdash e_1' : \tau'$*, and* $\cdot;\cdot \vdash e_2' : \tau'$ *we have* $f@\tau\ e_1\ e_2 \simeq f@\tau'\ e_1'\ e_2'$*.*

Proof. This proof proceeds similarly to Theorem 12.35, but we will choose to instantiate α at the candidate (τ, τ', R) where $e\ R\ e'$ always holds. (Again, this relation is closed under head expansion.) Unfolding definitions more rapidly, by Theorem 12.22 we have $f \sim f : \forall\alpha.\alpha \to \alpha \to \mathsf{ans}\ [\cdot]$. Instantiating at (τ, τ', R),

$$f@\tau \sim f@\tau' : \alpha \to \alpha \to \mathsf{ans}\ [(\tau, \tau', R)/\alpha]$$

and thus

$$e\ R\ e' \implies f@\tau\ e \sim f@\tau'\ e' : \alpha \to \mathsf{ans}\ [(\tau, \tau', R)/\alpha]$$

Plugging in $e_1\ R\ e_1'$ (which holds by the definition of R) we have

$$f@\tau\ e_1 \sim f@\tau'\ e_1' : \alpha \to \mathsf{ans}\ [(\tau, \tau', R)/\alpha]$$

and thus

$$e\ R\ e' \implies f@\tau\ e_1\ e \sim f@\tau'\ e_1'\ e' : \mathsf{ans}\ [(\tau, \tau', R)/\alpha]$$

Finally, plugging in $e_2\ R\ e_2'$ we have

$$f@\tau\ e_1\ e_2 \sim f@\tau'\ e_1'\ e_2' : \mathsf{ans}\ [(\tau, \tau', R)/\alpha]$$

and thus $f@\tau\ e_1\ e_2 \simeq f@\tau'\ e_1'\ e_2'$ as required. □

5 Deriving representation independence

Proving representation independence theorems is slightly more involved. Given two pack terms of existential type, we exhibit a bisimulation between them, use that bisimulation as the candidate relation to establish a logical equivalence, then use soundness to establish that the two pack terms are observationally equivalent.

These ideas scale directly to arbitrarily complex interfaces, but for the sake of this lecture we pick a rather simple "toggling" interface with an abstract type α, a starting state α, a toggling function $\alpha \to \alpha$, and a "toBool" function $\alpha \to \mathsf{bool}$ that converts the internal state to a boolean. We then implement two togglers, one using booleans ($\mathtt{impl}_1$) and the other using natural numbers ($\mathtt{impl}_2$).[14]

$$\mathtt{Toggler} := \exists\alpha.\alpha \times ((\alpha \to \alpha) \times (\alpha \to \mathsf{bool}))$$

$$\mathtt{impl}_1, \mathtt{impl}_2 : \mathtt{Toggler}$$

$$\mathtt{impl}_1 := \mathsf{pack}\ \langle \mathsf{bool}, (\mathsf{true}, (\mathsf{not}, \lambda x : \mathsf{bool}.x))\rangle\ \mathsf{as}\ \mathtt{Toggler}$$

$$\mathtt{impl}_2 := \mathsf{pack}\ \langle \mathsf{nat}, (\mathsf{zero}, (\lambda n : \mathsf{nat}.\mathsf{suc}(n), \mathsf{even?}))\rangle\ \mathsf{as}\ \mathtt{Toggler}$$

We then define the binary relation $e\ R\ e'$ between closed terms of type bool and closed terms of type nat to hold when $e \Downarrow \mathsf{true}$ and $\mathsf{even?}(e') \Downarrow \mathsf{true}$, or $e \Downarrow \mathsf{false}$ and $\mathsf{even?}(e') \Downarrow \mathsf{false}$. This relation is a Toggler *bisimulation between* $\mathtt{impl}_1$ *and* $\mathtt{impl}_2$ in the sense that the following four properties hold:

Lemma 12.38.

1. $(\mathsf{bool}, \mathsf{nat}, R)$ *is a candidate relation,*
2. $\mathsf{true}\ R\ \mathsf{zero}$*,*
3. *if* $e\ R\ e'$ *then* $\mathsf{not}(e)\ R\ \mathsf{suc}(e')$*, and*
4. *if* $e\ R\ e'$ *then either* $e \Downarrow \mathsf{true}$ *and* $\mathsf{even?}(e') \Downarrow \mathsf{true}$*, or* $e \Downarrow \mathsf{false}$ *and* $\mathsf{even?}(e') \Downarrow \mathsf{false}$*.*

Proof. (4) holds by definition; (1) and (2) are easily verified; (3) follows from the fact that $\mathsf{even?}(\mathsf{suc}(e')) = \mathsf{not}(\mathsf{even?}(e'))$ for all $e' : \mathsf{nat}$. □

Theorem 12.39 (Representation independence)**.** $\cdot;\cdot \vdash \mathtt{impl}_1 \cong \mathtt{impl}_2 : \mathtt{Toggler}$.

[14] We can either extend System F with bool, nat, and ×, or Church encode them. We assume some basic properties of closed logical equivalence at bool and × that will hold in either treatment.

Proof. By Theorem 12.34 it suffices to show $\mathtt{impl}_1 \sim \mathtt{impl}_2 : \mathtt{Toggler}\ [\cdot]$. Choosing the candidate $(\mathsf{bool}, \mathsf{nat}, R)$ by clause (1) of Lemma 12.38, it suffices to show

$$(\mathtt{true}, (\mathtt{not}, \lambda x : \mathsf{bool}.x)) \sim (\mathtt{zero}, (\lambda n : \mathsf{nat}.\mathsf{suc}(n), \mathtt{even?})) : \alpha \times ((\alpha \to \alpha) \times (\alpha \to \mathsf{bool}))\ [(\mathsf{bool}, \mathsf{nat}, R)/\alpha]$$

and hence that

- $\mathtt{true} \sim \mathtt{zero} : \alpha\ [(\mathsf{bool}, \mathsf{nat}, R)/\alpha]$,
- $\mathtt{not} \sim \lambda n : \mathsf{nat}.\mathsf{suc}(n) : \alpha \to \alpha\ [(\mathsf{bool}, \mathsf{nat}, R)/\alpha]$, and
- $\lambda x : \mathsf{bool}.x \sim \mathtt{even?} : \alpha \to \mathsf{bool}\ [(\mathsf{bool}, \mathsf{nat}, R)/\alpha]$.

These follow from clauses (2–4) of Lemma 12.38 respectively. □

In particular, for any client $\alpha\ \mathsf{ty}; x : \alpha \times ((\alpha \to \alpha) \times (\alpha \to \mathsf{bool})) \vdash e : \mathsf{ans}$ we have $(\mathtt{unpack}\ \langle \alpha, x \rangle = \mathtt{impl}_1\ \mathtt{in}\ e) \simeq (\mathtt{unpack}\ \langle \alpha, x \rangle = \mathtt{impl}_2\ \mathtt{in}\ e)$.

References

[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.