48 Parametricity

The main motivation for polymorphism is to enable more programs to be written—those that are "generic" in one or more types, such as the composition function given in Chapter 16. If a program *does not* depend on the choice of types, we can code it using polymorphism. Moreover, if we wish to insist that a program *cannot* depend on a choice of types, we demand that it be polymorphic. Thus, polymorphism can be used both to expand the collection of programs we may write and to limit the collection of programs that are permissible in a given context.

The restrictions imposed by polymorphic typing give rise to the experience that in a polymorphic functional language, if the types are correct, then the program is correct. Roughly speaking, if a function has a polymorphic type, then the strictures of type genericity cut down the set of programs with that type. Thus, if you have written a program with this type, it is more likely to be the one you intended!

The technical foundation for these remarks is called *parametricity*. The goal of this chapter is to give an account of parametricity for **F** under a call-by-name interpretation.

48.1 Overview

We will begin with an informal discussion of parametricity based on a "seat of the pants" understanding of the set of well-formed programs of a type.

Suppose that a function value f has the type $\forall(t.t \to t)$. What function could it be? When instantiated at a type τ it should evaluate to a function g of type $\tau \to \tau$ that, when further applied to a value v of type τ returns a value v' of type τ. Because f is polymorphic, g cannot depend on v, so v' must be v. In other words, g must be the identity function at type τ, and f must therefore be the *polymorphic identity*.

Suppose that f is a function of type $\forall(t.t)$. What function could it be? A moment's thought reveals that it cannot exist at all. For it must, when instantiated at a type τ, return a value of that type. But not every type has a value (including this one), so this is an impossible assignment. The only conclusion is that $\forall(t.t)$ is an *empty* type.

Let N be the type of polymorphic Church numerals introduced in Chapter 16, namely $\forall(t.t \to (t \to t) \to t)$. What are the values of this type? Given any type τ, and values $z : \tau$ and $s : \tau \to \tau$, the expression

$$f[\tau](z)(s)$$

https://doi.org/10.1017/CBO9781316576892.050 Published online by Cambridge University Press

must yield a value of type τ. Moreover, it must behave uniformly with respect to the choice of τ. What values could it yield? The only way to build a value of type τ is by using the element z and the function s passed to it. A moment's thought reveals that the application must amount to the n-fold composition

$$s(s(\ldots s(z)\ldots)).$$

That is, the elements of N are in one-to-one correspondence with the natural numbers.

48.2 Observational Equivalence

The definition of observational equivalence given in Chapters 46 and 47 is based on identifying a type of *answers* that are observable outcomes of complete programs. Values of function type are not regarded as answers, but are treated as "black boxes" with no internal structure, only input-output behavior. In **F**, however, there are no (closed) base types. Every type is either a function type or a polymorphic type, and hence no types suitable to serve as observable answers.

One way to manage this difficulty is to augment **F** with a base type of answers to serve as the observable outcomes of a computation. The only requirement is that this type have two elements that can be immediately distinguished from each other by evaluation. We may achieve this by enriching **F** with a base type **2** containing two constants, **tt** and **ff**, that serve as possible answers for a complete computation. A complete program is a closed expression of type **2**.

Kleene equality is defined for complete programs by requiring that $e \simeq e'$ iff either (a) $e \longmapsto^* \mathbf{tt}$ and $e' \longmapsto^* \mathbf{tt}$; or (b) $e \longmapsto^* \mathbf{ff}$ and $e' \longmapsto^* \mathbf{ff}$. This relation is an equivalence, and it is immediate that $\mathbf{tt} \not\simeq \mathbf{ff}$, because these are two distinct constants. As before, we say that a type-indexed family of equivalence relations between closed expressions of the same type is *consistent* if it implies Kleene equality at the answer type **2**.

To define observational equivalence, we must first define the concept of an expression context for **F** as an expression with a "hole" in it. More precisely, we may give an inductive definition of the judgment

$$\mathcal{C} : (\Delta; \Gamma \triangleright \tau) \rightsquigarrow (\Delta'; \Gamma' \triangleright \tau'),$$

which states that $\mathcal{C}$ is an expression context that, when filled with an expression $\Delta; \Gamma \vdash e : \tau$ yields an expression $\Delta'; \Gamma' \vdash \mathcal{C}\{e\} : \tau'$. (We leave the precise definition of this judgment, and the verification of its properties, as an exercise for the reader.)

Definition 48.1. *Two expressions of the same type are* observationally equivalent*, written* $\Delta; \Gamma \vdash e \cong e' : \tau$*, iff* $\mathcal{C}\{e\} \simeq \mathcal{C}\{e'\}$ *whenever* $\mathcal{C} : (\Delta; \Gamma \triangleright \tau) \rightsquigarrow (\emptyset; \emptyset \triangleright \mathbf{2})$.

Lemma 48.2. *Observational equivalence is the coarsest consistent congruence.*

Proof Essentially the same as the the proof of Theorem 46.6. □

https://doi.org/10.1017/CBO9781316576892.050 Published online by Cambridge University Press

Lemma 48.3.

1. *If* $\Delta, t; \Gamma \vdash e \cong e' : \tau$ *and* τ_0 *type, then* $\Delta; [\tau_0/t]\Gamma \vdash [\tau_0/t]e \cong [\tau_0/t]e' : [\tau_0/t]\tau$.
2. *If* $\emptyset; \Gamma, x : \tau_0 \vdash e \cong e' : \tau$ *and* $d : \tau_0$, *then* $\emptyset; \Gamma \vdash [d/x]e \cong [d/x]e' : \tau$. *Moreover, if* $d \cong_{\tau_0} d'$, *then* $\emptyset; \Gamma \vdash [d/x]e \cong [d'/x]e : \tau$ *and* $\emptyset; \Gamma \vdash [d/x]e' \cong [d'/x]e' : \tau$.

Proof 1. Let $\mathcal{C} : (\Delta; [\tau_0/t]\Gamma \triangleright [\tau_0/t]\tau) \rightsquigarrow (\emptyset \triangleright \mathbf{2})$ be a program context. We are to show that

$$\mathcal{C}\{[\tau_0/t]e\} \simeq \mathcal{C}\{[\tau_0/t]e'\}.$$

Because $\mathcal{C}$ is closed, this is equivalent to

$$[\tau_0/t]\mathcal{C}\{e\} \simeq [\tau_0/t]\mathcal{C}\{e'\}.$$

Let $\mathcal{C}'$ be the context $\Lambda(t)\,\mathcal{C}\{\circ\}[\tau_0]$, and observe that

$$\mathcal{C}' : (\Delta, t; \Gamma \triangleright \tau) \rightsquigarrow (\emptyset \triangleright \mathbf{2}).$$

Therefore, from the assumption,

$$\mathcal{C}'\{e\} \simeq \mathcal{C}'\{e'\}.$$

But $\mathcal{C}'\{e\} \simeq [\tau_0/t]\mathcal{C}\{e\}$, and $\mathcal{C}'\{e'\} \simeq [\tau_0/t]\mathcal{C}\{e'\}$, from which the result follows.

2. By an argument similar to that for Lemma 46.7. □

48.3 Logical Equivalence

In this section, we introduce a form of logical equivalence that captures the informal concept of parametricity, and also provides a characterization of observational equivalence. This characterization will permit us to derive properties of observational equivalence of polymorphic programs of the kind suggested earlier.

The definition of logical equivalence for **F** is somewhat more complex than for **T**. The main idea is to define logical equivalence for a polymorphic type $\forall(t.\tau)$ to satisfy a very strong condition that captures the essence of parametricity. As a first approximation, we might say that two expressions e and e' of this type should be logically equivalent if they are logically equivalent for "all possible" interpretations of the type t. More precisely, we might require that $e[\rho]$ be related to $e'[\rho]$ at type $[\rho/t]\tau$, for any choice of type ρ. But this runs into two problems, one technical, the other conceptual. The same device will be used to solve both problems.

The technical problem stems from impredicativity. In Chapter 46, logical equivalence is defined by induction on the structure of types. But when polymorphism is impredicative, the type $[\rho/t]\tau$ might well be larger than $\forall(t.\tau)$. At the very least, we would have to justify the definition of logical equivalence on some other grounds, but no criterion appears to be available. The conceptual problem is that, even if we could make sense of the definition

https://doi.org/10.1017/CBO9781316576892.050 Published online by Cambridge University Press

of logical equivalence, it would be too restrictive. For such a definition amounts to saying that the unknown type t is interpreted as logical equivalence at whatever type it is when instantiated. To obtain useful parametricity results, we shall ask for much more than this. What we shall do is to consider *separately* instances of e and e' by types ρ and ρ', and treat the type variable t as standing for *any relation* (of some form) between ρ and ρ'. We may suspect that this is asking too much: perhaps logical equivalence is the *empty* relation. Surprisingly, this is not the case, and indeed it is this very feature of the definition that we shall exploit to derive parametricity results about the language.

To manage both of these problems, we will consider a generalization of logical equivalence that is parameterized by a relational interpretation of the free type variables of its classifier. The parameters determine a separate binding for each free type variable in the classifier for each side of the equation, with the discrepancy being mediated by a specified relation between them. Thus, related expressions need not have the same type, with the differences between them mediated by the given relation.

We will restrict attention to a certain collection of "admissible" binary relations between closed expressions. The conditions are imposed to ensure that logical equivalence and observational equivalence coincide.

Definition 48.4 (Admissibility)**.** *A relation R between expressions of types ρ and ρ' is* admissible*, written $R : \rho \leftrightarrow \rho'$, iff it satisfies two requirements:*

1. *Respect for observational equivalence: if $R(e, e')$ and $d \cong_\rho e$ and $d' \cong_{\rho'} e'$, then $R(d, d')$.*
2. *Closure under converse evaluation: if $R(e, e')$, then if $d \longmapsto e$, then $R(d, e')$ and if $d' \longmapsto e'$, then $R(e, d')$.*

Closure under converse evaluation is a consequence of respect for observational equivalence, but we are not yet in a position to establish this fact.

The judgment $\delta : \Delta$ states that δ is a *type substitution* that assigns a closed type to each type variable $t \in \Delta$. A type substitution δ induces a substitution function $\hat{\delta}$ on types given by the equation

$$\hat{\delta}(\tau) = [\delta(t_1), \ldots, \delta(t_n)/t_1, \ldots, t_n]\tau,$$

and similarly for expressions. Substitution is extended to contexts point-wise by defining $\hat{\delta}(\Gamma)(x) = \hat{\delta}(\Gamma(x))$ for each $x \in dom(\Gamma)$.

Let δ and δ' be two type substitutions of closed types to the type variables in Δ. An *admissible relation assignment* η between δ and δ' is an assignment of an admissible relation $\eta(t) : \delta(t) \leftrightarrow \delta'(t)$ to each $t \in \Delta$. The judgment $\eta : \delta \leftrightarrow \delta'$ states that η is an admissible relation assignment between δ and δ'.

Logical equivalence is defined in terms of its generalization, called *parametric logical equivalence*, written $e \sim_\tau e' \, [\eta : \delta \leftrightarrow \delta']$, defined as follows.

https://doi.org/10.1017/CBO9781316576892.050 Published online by Cambridge University Press

Definition 48.5 (Parametric Logical Equivalence)**.** *The relation* $e \sim_\tau e' \ [\eta : \delta \leftrightarrow \delta']$ *is defined by induction on the structure of* τ *by the following conditions:*

$$
\begin{array}{lll}
e \sim_t e' \ [\eta : \delta \leftrightarrow \delta'] & \textit{iff} & \eta(t)(e, e') \\
e \sim_{\mathbf{2}} e' \ [\eta : \delta \leftrightarrow \delta'] & \textit{iff} & e \simeq e' \\
e \sim_{\tau_1 \to \tau_2} e' \ [\eta : \delta \leftrightarrow \delta'] & \textit{iff} & e_1 \sim_{\tau_1} e_1' \ [\eta : \delta \leftrightarrow \delta'] \textit{ implies} \\
 & & e(e_1) \sim_{\tau_2} e'(e_1') \ [\eta : \delta \leftrightarrow \delta'] \\
e \sim_{\forall(t.\tau)} e' \ [\eta : \delta \leftrightarrow \delta'] & \textit{iff} & \textit{for every } \rho,\ \rho',\ \textit{and every admissible } R : \rho \leftrightarrow \rho', \\
 & & e[\rho] \sim_\tau e'[\rho'] \ [\eta \otimes t \hookrightarrow R : \delta \otimes t \hookrightarrow \rho \leftrightarrow \delta' \otimes t \hookrightarrow \rho']
\end{array}
$$

Logical equivalence is defined in terms of parametric logical equivalence by considering all possible interpretations of its free type- and expression variables. An *expression substitution* γ for a context Γ, written $\gamma : \Gamma$, is a substitution of a closed expression $\gamma(x) : \Gamma(x)$ to each variable $x \in dom(\Gamma)$. An expression substitution $\gamma : \Gamma$ induces a substitution function, $\hat{\gamma}$, defined by the equation

$$\hat{\gamma}(e) = [\gamma(x_1), \ldots, \gamma(x_n)/x_1, \ldots, x_n]e,$$

where the domain of Γ consists of the variables $x_1, \ldots, x_n$. The relation $\gamma \sim_\Gamma \gamma' \ [\eta : \delta \leftrightarrow \delta']$ is defined to hold iff $dom(\gamma) = dom(\gamma') = dom(\Gamma)$, and $\gamma(x) \sim_{\Gamma(x)} \gamma'(x) \ [\eta : \delta \leftrightarrow \delta']$ for every variable x in their common domain.

Definition 48.6 (Logical Equivalence)**.** *The expressions* $\Delta; \Gamma \vdash e : \tau$ *and* $\Delta; \Gamma \vdash e' : \tau$ *are* logically equivalent*, written* $\Delta; \Gamma \vdash e \sim e' : \tau$ *iff, for every assignment* δ *and* δ' *of closed types to type variables in* Δ*, and every admissible relation assignment* $\eta : \delta \leftrightarrow \delta'$*, if* $\gamma \sim_\Gamma \gamma' \ [\eta : \delta \leftrightarrow \delta']$*, then* $\hat{\gamma}(\hat{\delta}(e)) \sim_\tau \widehat{\gamma'}(\widehat{\delta'}(e')) \ [\eta : \delta \leftrightarrow \delta']$.

When e, e', and τ are closed, this definition states that $e \sim_\tau e'$ iff $e \sim_\tau e' \ [\emptyset : \emptyset \leftrightarrow \emptyset]$, so that logical equivalence is indeed a special case of its generalization.

Lemma 48.7 (Closure under Converse Evaluation)**.** *Suppose that* $e \sim_\tau e' \ [\eta : \delta \leftrightarrow \delta']$*. If* $d \longmapsto e$*, then* $d \sim_\tau e'$*, and if* $d' \longmapsto e'$*, then* $e \sim_\tau d'$.

Proof By induction on the structure of τ. When $\tau = t$, the result holds by the definition of admissibility. Otherwise, the result follows by induction, making use of the definition of the transition relation for applications and type applications. □

Lemma 48.8 (Respect for Observational Equivalence)**.** *Suppose that* $e \sim_\tau e' \ [\eta : \delta \leftrightarrow \delta']$*. If* $d \cong_{\hat{\delta}(\tau)} e$ *and* $d' \cong_{\widehat{\delta'}(\tau)} e'$*, then* $d \sim_\tau d' \ [\eta : \delta \leftrightarrow \delta']$.

Proof By induction on the structure of τ, relying on the definition of admissibility, and the congruence property of observational equivalence. For example, if $\tau = \forall(t.\tau_2)$, then we are to show that for every admissible $R : \rho \leftrightarrow \rho'$,

$$d[\rho] \sim_{\tau_2} d'[\rho'] \ [\eta \otimes t \hookrightarrow R : \delta \otimes t \hookrightarrow \rho \leftrightarrow \delta' \otimes t \hookrightarrow \rho'].$$

https://doi.org/10.1017/CBO9781316576892.050 Published online by Cambridge University Press

Because observational equivalence is a congruence, we have $d[\rho] \cong_{[\rho/t]\hat{\delta}(\tau_2)} e[\rho]$, and $d'[\rho'] \cong_{[\rho'/t]\widehat{\delta'}(\tau_2)} e'[\rho]$. It follows that

$$e[\rho] \sim_{\tau_2} e'[\rho']\ [\eta \otimes t \hookrightarrow R : \delta \otimes t \hookrightarrow \rho \leftrightarrow \delta' \otimes t \hookrightarrow \rho'],$$

from which the result follows by induction. □

Corollary 48.9. *The relation $e \sim_\tau e'\ [\eta : \delta \leftrightarrow \delta']$ is an admissible relation between closed types $\hat{\delta}(\tau)$ and $\widehat{\delta'}(\tau)$.*

Proof By Lemmas 48.7 and 48.8. □

Corollary 48.10. *If $\Delta; \Gamma \vdash e \sim e' : \tau$, and $\Delta; \Gamma \vdash d \cong e : \tau$ and $\Delta; \Gamma \vdash d' \cong e' : \tau$, then $\Delta; \Gamma \vdash d \sim d' : \tau$.*

Proof By Lemma 48.3 and Corollary 48.9. □

Lemma 48.11 (Compositionality)**.** *Let $R : \hat{\delta}(\rho) \leftrightarrow \widehat{\delta'}(\rho)$ be the relational interpretation of some type ρ, which is to say $R(d, d')$ holds iff $d \sim_\rho d'\ [\eta : \delta \leftrightarrow \delta']$. Then $e \sim_{[\rho/t]\tau} e'\ [\eta : \delta \leftrightarrow \delta']$ if, and only if,*

$$e \sim_\tau e'\ [\eta \otimes t \hookrightarrow R : \delta \otimes t \hookrightarrow \hat{\delta}(\rho) \leftrightarrow \delta' \otimes t \hookrightarrow \widehat{\delta'}(\rho)].$$

Proof By induction on the structure of τ. When $\tau = t$, the result is immediate from the choice of the relation R. When $\tau = t' \neq t$, the result follows from Definition 48.5. When $\tau = \tau_1 \to \tau_2$, the result follows by induction, using Definition 48.5. Similarly, when or $\tau = \forall(u.\tau_1)$, the result follows by induction, noting that we may assume, without loss of generality, that $u \neq t$ and $u \notin \rho$. □

Despite the strong conditions on polymorphic types, logical equivalence is not too restrictive—every expression satisfies its constraints. This result is often called the *parametricity theorem* or the *abstraction theorem*:

Theorem 48.12 (Parametricity)**.** *If $\Delta; \Gamma \vdash e : \tau$, then $\Delta; \Gamma \vdash e \sim e : \tau$.*

Proof By rule induction on the statics of **F** given by rules (16.2).

We consider two representative cases here.

Rule (16.2d) Suppose $\delta : \Delta, \delta' : \Delta, \eta : \delta \leftrightarrow \delta'$, and $\gamma \sim_\Gamma \gamma'\ [\eta : \delta \leftrightarrow \delta']$. By induction we have that for all ρ, ρ', and admissible $R : \rho \leftrightarrow \rho'$,

$$[\rho/t]\hat{\gamma}(\hat{\delta}(e)) \sim_\tau [\rho'/t]\widehat{\gamma'}(\widehat{\delta'}(e))\ [\eta_* : \delta_* \leftrightarrow \delta'_*],$$

where $\eta_* = \eta \otimes t \hookrightarrow R$, $\delta_* = \delta \otimes t \hookrightarrow \rho$, and $\delta'_* = \delta' \otimes t \hookrightarrow \rho'$. Because

$$\Lambda(t)\, \hat{\gamma}(\hat{\delta}(e))[\rho] \longmapsto^* [\rho/t]\hat{\gamma}(\hat{\delta}(e))$$

https://doi.org/10.1017/CBO9781316576892.050 Published online by Cambridge University Press

and

$$\Lambda(t)\,\widehat{\gamma'}(\widehat{\delta'}(e))[\rho'] \longmapsto^* [\rho'/t]\widehat{\gamma'}(\widehat{\delta'}(e)),$$

the result follows by Lemma 48.7.

Rule (16.2e) Suppose $\delta : \Delta, \delta' : \Delta, \eta : \delta \leftrightarrow \delta'$, and $\gamma \sim_\Gamma \gamma'\ [\eta : \delta \leftrightarrow \delta']$. By induction we have

$$\hat{\gamma}(\hat{\delta}(e)) \sim_{\forall(t.\tau)} \widehat{\gamma'}(\widehat{\delta'}(e))\ [\eta : \delta \leftrightarrow \delta']$$

Let $\hat{\rho} = \hat{\delta}(\rho)$ and $\hat{\rho'} = \widehat{\delta'}(\rho)$. Define the relation $R : \hat{\rho} \leftrightarrow \hat{\rho'}$ by $R(d, d')$ iff $d \sim_\rho d'\ [\eta : \delta \leftrightarrow \delta']$. By Corollary 48.9, this relation is admissible.

By the definition of logical equivalence at polymorphic types, we obtain

$$\hat{\gamma}(\hat{\delta}(e))[\hat{\rho}] \sim_\tau \widehat{\gamma'}(\widehat{\delta'}(e))[\hat{\rho'}]\ [\eta \otimes t \hookrightarrow R : \delta \otimes t \hookrightarrow \hat{\rho} \leftrightarrow \delta' \otimes t \hookrightarrow \hat{\rho'}].$$

By Lemma 48.11

$$\hat{\gamma}(\hat{\delta}(e))[\hat{\rho}] \sim_{[\rho/t]\tau} \widehat{\gamma'}(\widehat{\delta'}(e))[\hat{\rho'}]\ [\eta : \delta \leftrightarrow \delta']$$

But

$$\hat{\gamma}(\hat{\delta}(e))[\hat{\rho}] = \hat{\gamma}(\hat{\delta}(e))[\hat{\delta}(\rho)] \tag{48.1}$$
$$= \hat{\gamma}(\hat{\delta}(e[\rho])), \tag{48.2}$$

and similarly

$$\widehat{\gamma'}(\widehat{\delta'}(e))[\hat{\rho'}] = \widehat{\gamma'}(\widehat{\delta'}(e))[\widehat{\delta'}(\rho)] \tag{48.3}$$
$$= \widehat{\gamma'}(\widehat{\delta'}(e[\rho])), \tag{48.4}$$

from which the result follows. □

Corollary 48.13. *If* $\Delta; \Gamma \vdash e \cong e' : \tau$, *then* $\Delta; \Gamma \vdash e \sim e' : \tau$.

Proof By Theorem 48.12 $\Delta; \Gamma \vdash e \sim e : \tau$, and hence by Corollary 48.10, $\Delta; \Gamma \vdash e \sim e' : \tau$. □

Lemma 48.14 (Congruence)**.** *If* $\Delta; \Gamma \vdash e \sim e' : \tau$ *and* $\mathcal{C} : (\Delta; \Gamma \triangleright \tau) \rightsquigarrow (\Delta'; \Gamma' \triangleright \tau')$, *then* $\Delta'; \Gamma' \vdash \mathcal{C}\{e\} \sim \mathcal{C}\{e'\} : \tau'$.

Proof By induction on the structure of $\mathcal{C}$, following along very similar lines to the proof of Theorem 48.12. □

Lemma 48.15 (Consistency)**.** *Logical equivalence is consistent.*

Proof Follows from the definition of logical equivalence. □

Corollary 48.16. *If* $\Delta; \Gamma \vdash e \sim e' : \tau$, *then* $\Delta; \Gamma \vdash e \cong e' : \tau$.

https://doi.org/10.1017/CBO9781316576892.050 Published online by Cambridge University Press

Proof By Lemma 48.15, logical equivalence is consistent, and by Lemma 48.14, it is a congruence, and hence is contained in observational equivalence. □

Corollary 48.17. *Logical and observational equivalence coincide.*

Proof By Corollaries 48.13 and 48.16. □

If $d : \tau$ and $d \longmapsto e$, then $d \sim_\tau e$, and hence by Corollary 48.16, $d \cong_\tau e$. Therefore, if a relation respects observational equivalence, it must also be closed under converse evaluation. The second condition on admissibility is superfluous, now that we have established the coincidence of logical and observational equivalence.

Corollary 48.18 (Extensionality)**.**

1. $e \cong_{\tau_1 \to \tau_2} e'$ *iff for all* $e_1 : \tau_1$, $e(e_1) \cong_{\tau_2} e'(e_1)$.
2. $e \cong_{\forall(t.\tau)} e'$ *iff for all* ρ, $e[\rho] \cong_{[\rho/t]\tau} e'[\rho]$.

Proof The forward direction is immediate in both cases, because observational equivalence is a congruence, by definition. The backward direction is proved similarly in both cases, by appeal to Theorem 48.12. In the first case, by Corollary 48.17 it suffices to show that $e \sim_{\tau_1 \to \tau_2} e'$. To this end, suppose that $e_1 \sim_{\tau_1} e_1'$. We are to show that $e(e_1) \sim_{\tau_2} e'(e_1')$. By the assumption, we have $e(e_1') \cong_{\tau_2} e'(e_1')$. By parametricity, we have $e \sim_{\tau_1 \to \tau_2} e$, and hence $e(e_1) \sim_{\tau_2} e(e_1')$. The result then follows by Lemma 48.8. In the second case, by Corollary 48.17 it is sufficient to show that $e \sim_{\forall(t.\tau)} e'$. Suppose that $R : \rho \leftrightarrow \rho'$ for some closed types ρ and ρ'. It suffices to show that $e[\rho] \sim_\tau e'[\rho']\ [\eta : \delta \leftrightarrow \delta']$, where $\eta(t) = R$, $\delta(t) = \rho$, and $\delta'(t) = \rho'$. By the assumption, we have $e[\rho'] \cong_{[\rho'/t]\tau} e'[\rho']$. By parametricity $e \sim_{\forall(t.\tau)} e$, and hence $e[\rho] \sim_\tau e'[\rho']\ [\eta : \delta \leftrightarrow \delta']$. The result then follows by Lemma 48.8. □

Lemma 48.19 (Identity Extension)**.** *Let* $\eta : \delta \leftrightarrow \delta$ *be such that* $\eta(t)$ *is observational equivalence at type* $\delta(t)$ *for each* $t \in dom(\delta)$*. Then* $e \sim_\tau e'\ [\eta : \delta \leftrightarrow \delta]$ *iff* $e \cong_{\delta(\tau)} e'$.

Proof The backward direction follows from Theorem 48.12 and respect for observational equivalence. The forward direction is proved by induction on the structure of τ, appealing to Corollary 48.18 to establish observational equivalence at function and polymorphic types. □

48.4 Parametricity Properties

The parametricity theorem enables us to deduce properties of expressions of **F** that hold solely because of their type. The stringencies of parametricity ensure that a polymorphic

https://doi.org/10.1017/CBO9781316576892.050 Published online by Cambridge University Press

type has very few inhabitants. For example, we may prove that *every* expression of type $\forall(t.t \to t)$ behaves like the identity function.

Theorem 48.20. *Let* $e : \forall(t.t \to t)$ *be arbitrary, and let* id *be* $\Lambda(t)\,\lambda(x:t)\,x$. *Then* $e \cong_{\forall(t.t\to t)} id$.

Proof By Corollary 48.17 it is sufficient to show that $e \sim_{\forall(t.t\to t)} id$. Let ρ and ρ' be arbitrary closed types, let $R : \rho \leftrightarrow \rho'$ be an admissible relation, and suppose that $e_0\ R\ e_0'$. We are to show

$$e[\rho](e_0)\ R\ id[\rho](e_0'),$$

which, given the definition of id and closure under converse evaluation, is to say

$$e[\rho](e_0)\ R\ e_0'.$$

It suffices to show that $e[\rho](e_0) \cong_\rho e_0$, for then the result follows by the admissibility of R and the assumption $e_0\ R\ e_0'$.

By Theorem 48.12, we have $e \sim_{\forall(t.t\to t)} e$. Let the relation $S : \rho \leftrightarrow \rho$ be defined by $d\ S\ d'$ iff $d \cong_\rho e_0$ and $d' \cong_\rho e_0$. This relation is clearly admissible, and we have $e_0\ S\ e_0$. It follows that

$$e[\rho](e_0)\ S\ e[\rho](e_0),$$

and so, by the definition of the relation S, $e[\rho](e_0) \cong_\rho e_0$. □

In Chapter 16, we showed that product, sum, and natural numbers types are all definable in **F**. The proof of definability in each case consisted of showing that the type and its associated introduction and elimination forms are encodable in **F**. The encodings are correct in the (weak) sense that the dynamics of these constructs as given in the earlier chapters is derivable from the dynamics of **F** via these definitions. By taking advantage of parametricity, we may extend these results to obtain a strong correspondence between these types and their encodings.

As a first example, let us consider the representation of the unit type, `unit`, in **F**, as defined in Chapter 16 by the following equations:

$$\begin{aligned} \texttt{unit} &= \forall(r.r \to r) \\ \langle\rangle &= \Lambda(r)\,\lambda(x:r)\,x \end{aligned}$$

It is easy to see that $\langle\rangle : \texttt{unit}$ according to these definitions. But this says that the type `unit` is inhabited (has an element). What we would like to know is that, up to observational equivalence, the expression $\langle\rangle$ is the *only* element of that type. But this is the content of Theorem 48.20. We say that the type `unit` is *strongly definable* within **F**.

https://doi.org/10.1017/CBO9781316576892.050 Published online by Cambridge University Press

Continuing in this vein, let us examine the definition of the binary product type in **F**, also given in Chapter 16:

$$\begin{aligned}
\tau_1 \times \tau_2 &= \forall(r.(\tau_1 \to \tau_2 \to r) \to r) \\
\langle e_1, e_2 \rangle &= \Lambda(r)\,\lambda\,(x : \tau_1 \to \tau_2 \to r)\,x(e_1)(e_2) \\
e \cdot \mathtt{l} &= e[\tau_1](\lambda\,(x : \tau_1)\,\lambda\,(y : \tau_2)\,x) \\
e \cdot \mathtt{r} &= e[\tau_2](\lambda\,(x : \tau_1)\,\lambda\,(y : \tau_2)\,y)
\end{aligned}$$

It is easy to check that $\langle e_1, e_2 \rangle \cdot \mathtt{l} \cong_{\tau_1} e_1$ and $\langle e_1, e_2 \rangle \cdot \mathtt{r} \cong_{\tau_2} e_2$ by a direct calculation.

We wish to show that the ordered pair, as defined above, is the unique such expression, and hence that Cartesian products are strongly definable in **F**. We will make use of a lemma governing the behavior of the elements of the product type whose proof relies on Theorem 48.12.

Lemma 48.21. *If $e : \tau_1 \times \tau_2$, then $e \cong_{\tau_1 \times \tau_2} \langle e_1, e_2 \rangle$ for some $e_1 : \tau_1$ and $e_2 : \tau_2$.*

Proof Expanding the definitions of pairing and the product type, and applying Corollary 48.17, we let ρ and ρ' be arbitrary closed types, and let $R : \rho \leftrightarrow \rho'$ be an admissible relation between them. Suppose further that

$$h \sim_{\tau_1 \to \tau_2 \to t} h'\ [\eta : \delta \leftrightarrow \delta'],$$

where $\eta(t) = R$, $\delta(t) = \rho$, and $\delta'(t) = \rho'$ (and each is undefined on $t' \neq t$). We are to show that for some $e_1 : \tau_1$ and $e_2 : \tau_2$,

$$e[\rho](h) \sim_t h'(e_1)(e_2)\ [\eta : \delta \leftrightarrow \delta'],$$

which is to say

$$e[\rho](h)\ R\ h'(e_1)(e_2).$$

Now by Theorem 48.12 we have $e \sim_{\tau_1 \times \tau_2} e$. Define the relation $S : \rho \leftrightarrow \rho'$ by $d\ S\ d'$ iff the following conditions are satisfied:

1. $d \cong_\rho h(d_1)(d_2)$ for some $d_1 : \tau_1$ and $d_2 : \tau_2$;
2. $d' \cong_{\rho'} h'(d_1')(d_2')$ for some $d_1' : \tau_1$ and $d_2' : \tau_2$;
3. $d\ R\ d'$.

This relation is clearly admissible. Noting that

$$h \sim_{\tau_1 \to \tau_2 \to t} h'\ [\eta' : \delta \leftrightarrow \delta'],$$

where $\eta'(t) = S$ and $\eta'(t')$ is undefined for $t' \neq t$, we conclude that $e[\rho](h)\ S\ e[\rho'](h')$, and hence

$$e[\rho](h)\ R\ h'(d_1')(d_2'),$$

as required. □

https://doi.org/10.1017/CBO9781316576892.050 Published online by Cambridge University Press

Now suppose that $e : \tau_1 \times \tau_2$ is such that $e \cdot \mathtt{l} \cong_{\tau_1} e_1$ and $e \cdot \mathtt{r} \cong_{\tau_2} e_2$. We wish to show that $e \cong_{\tau_1 \times \tau_2} \langle e_1, e_2 \rangle$. From Lemma 48.21 it follows that $e \cong_{\tau_1 \times \tau_2} \langle e \cdot \mathtt{l}, e \cdot \mathtt{r} \rangle$ by congruence and direct calculation. Hence, by congruence we have $e \cong_{\tau_1 \times \tau_2} \langle e_1, e_2 \rangle$.

By a similar line of reasoning, we may show that the Church encoding of the natural numbers given in Chapter 16 strongly defines the natural numbers in that the following properties hold:

1. $\mathtt{iter}\,\mathtt{z}\,\{\mathtt{z} \hookrightarrow e_0 \mid \mathtt{s}(x) \hookrightarrow e_1\} \cong_\rho e_0$.
2. $\mathtt{iter}\,\mathtt{s}(e)\,\{\mathtt{z} \hookrightarrow e_0 \mid \mathtt{s}(x) \hookrightarrow e_1\} \cong_\rho [\mathtt{iter}\,e\,\{\mathtt{z} \hookrightarrow e_0 \mid \mathtt{s}(x) \hookrightarrow e_1\}/x]e_1$.
3. Suppose that $x : \mathtt{nat} \vdash r(x) : \rho$. If
   (a) $r(\mathtt{z}) \cong_\rho e_0$, and
   (b) $r(\mathtt{s}(e)) \cong_\rho [r(e)/x]e_1$,
   then for every $e : \mathtt{nat}$, $r(e) \cong_\rho \mathtt{iter}\,e\,\{\mathtt{z} \hookrightarrow e_0 \mid \mathtt{s}(x) \hookrightarrow e_1\}$.

The first two equations, which constitute weak definability, are easily established by calculation, using the definitions given in Chapter 16. The third property, the unicity of the iterator, is proved using parametricity by showing that every closed expression of type $\mathtt{nat}$ is observationally equivalent to a numeral $\overline{n}$. We then argue for unicity of the iterator by mathematical induction on $n \geq 0$.

Lemma 48.22. *If $e : \mathtt{nat}$, then either $e \cong_{\mathtt{nat}} \mathtt{z}$, or there exists $e' : \mathtt{nat}$ such that $e \cong_{\mathtt{nat}} \mathtt{s}(e')$. Consequently, there exists $n \geq 0$ such that $e \cong_{\mathtt{nat}} \overline{n}$.*

Proof By Theorem 48.12, we have $e \sim_{\mathtt{nat}} e$. Define the relation $R : \mathtt{nat} \leftrightarrow \mathtt{nat}$ to be the strongest relation such that $d\ R\ d'$ iff either $d \cong_{\mathtt{nat}} \mathtt{z}$ and $d' \cong_{\mathtt{nat}} \mathtt{z}$, or $d \cong_{\mathtt{nat}} \mathtt{s}(d_1)$ and $d' \cong_{\mathtt{nat}} \mathtt{s}(d_1')$ and $d_1\ R\ d_1'$. It is easy to see that $\mathtt{z}\ R\ \mathtt{z}$, and if $e\ R\ e'$, then $\mathtt{s}(e)\ R\ \mathtt{s}(e')$. Letting $\mathtt{zero} = \mathtt{z}$ and $\mathtt{succ} = \lambda\,(x : \mathtt{nat})\,\mathtt{s}(x)$, we have

$$e[\mathtt{nat}](\mathtt{zero})(\mathtt{succ})\ R\ e[\mathtt{nat}](\mathtt{zero})(\mathtt{succ}).$$

The result follows by the induction principle arising from the definition of R as the strongest relation satisfying its defining conditions. □

48.5 Representation Independence, Revisited

In Section 17.4, we discussed the property of *representation independence* for abstract types. If two implementations of an abstract type are "similar," then the client behavior is not affected by replacing one for the other. The crux of the matter is the definition of similarity of two implementations. Informally, two implementations of an abstract type are similar if there is a relation R between their representation types that is *preserved* by the operations of the type. The relation R may be thought of as expressing the "equivalence"

https://doi.org/10.1017/CBO9781316576892.050 Published online by Cambridge University Press

of the two representations; checking that each operation preserves R amounts to checking that the result of performing that operation on equivalent representations yields equivalent results.

As an example, we argued in Section 17.4 that two implementations of a queue abstraction are similar. The two representations of queues are related by a relation R such that $q\ R\ (b,\ f)$ iff q is b followed by the reversal of f. When then argued that the operations preserve this relationship, and then claimed, without proof, that the behavior of the client would not be disrupted by changing one implementation to the other.

The proof of this claim relies on parametricity, as may be seen by considering the definability of existential types in **F** given in Section 17.3. According to that definition, the client, e, of an abstract type $\exists(t.\tau)$ is a polymorphic function of type $\forall(t.\tau \to \tau_2)$, where τ_2, the result type of the computation, does not involve the type variable t. Being polymorphic, the client enjoys the parametricity property given by Theorem 48.12. Specifically, suppose that ρ_1 and ρ_2 are two closed representation types and that $R : \rho_1 \leftrightarrow \rho_2$ is an admissible relation between them. For example, in the case of the queue abstraction, ρ_1 is the type of lists of elements of the queue, ρ_2 is the type of a pair of lists of elements, and R is the relation given above. Suppose further that $e_1 : [\rho_1/t]\tau$ and $e_2 : [\rho_2/t]\tau$ are two implementations of the operations such that

$$e_1 \sim_\tau e_2\ [\eta : \delta_1 \leftrightarrow \delta_2], \tag{48.5}$$

where $\eta(t) = R$, $\delta_1(t) = \rho_1$, and $\delta_2(t) = \rho_2$. In the case of the queues example, the expression e_1 is the implementation of the queue operations in terms of lists, and the e_2 is the implementation in terms of pairs of lists described earlier. Condition (48.5) states that the two implementations are similar in that they preserve the relation R between the representation types. By Theorem 48.12, it follows that the client e satisfies

$$e \sim_{\tau_2} e\ [\eta : \delta_1 \leftrightarrow \delta_2].$$

But because τ_2 is a closed type (in particular, does not involve t), this is equivalent to

$$e \sim_{\tau_2} e\ [\emptyset : \emptyset \leftrightarrow \emptyset].$$

But then by Lemma 48.19 we have

$$e[\rho_1](e_1) \cong_{\tau_2} e[\rho_2](e_2).$$

That is, the client behavior is not affected by the change of representation.

48.6 Notes

The concept of parametricity is latent in the proof of normalization for System **F** (Girard, 1972). Reynolds (1983), though technically flawed due to its reliance on a (non-existent) set-theoretic model of polymorphism, emphasizes the centrality of logical equivalence for characterizing equality of polymorphic programs. The application of parametricity

https://doi.org/10.1017/CBO9781316576892.050 Published online by Cambridge University Press

to representation independence was suggested by Reynolds and developed for existential types by Mitchell (1986) and Pitts (1998). The extension of System **F** with a "positive" (inductively defined) observable type appears to be needed to even define observational equivalence, but this point seems not to have been made elsewhere in the literature.

https://doi.org/10.1017/CBO9781316576892.050 Published online by Cambridge University Press