2024 OPLSS: Foundations of Programming Languages - Lecture 4

Lecturer: Paul Downen
Note Takers: Aaryan Patel, Adriano Corbelino II, Anders Thuné, Thomas Porter

June 2024

1 Typed Observational Equivalence

Recall untyped observational equivalence was not powerful enough to validate some equations that we would hope to hold, such as $\lambda x.\ not\ (not\ x) = \lambda x.\ x$ or $and\ x\ y = and\ y\ x$. This is because in contexts that apply these functions to non-boolean terms, the underlying conditional will be stuck, and the sides of the equations cannot be reduced to the same value. We would like a way to exclude these ill-typed counterexamples to observational indistinguishability. This motivates the following definition of typed observational equivalence.

Definition 1.1 (Typed Observational/Contextual Equivalence)**.** *Two terms M and N are observationally equivalent in environment Γ at type A, written $\Gamma \vdash M \approx N : A$, if and only if:*

- $\Gamma \vdash M, N : A$, *and*
- *for all $C \in$ Contexts such that $\cdot \vdash C[M] : \mathsf{bool}$ and $\cdot \vdash C[N] : \mathsf{bool}$, there exist values v, w with $C[M] \mapsto\!\!\!\!\twoheadrightarrow_\beta v \overset{\mathsf{bool}}{\sim} w \ {}_\beta\!\!\!\!\twoheadleftarrow\!\!\!\mapsto C[N]$,*

where $\overset{\mathsf{bool}}{\sim}$ is defined by the rules true $\overset{\mathsf{bool}}{\sim}$ true and false $\overset{\mathsf{bool}}{\sim}$ false.

Two terms of type A are observationally equivalent at type A if observing them in all closed and *well-typed* contexts of type bool results in the same value. The choice of bool here was arbitrary in a sense: the key property we want is to have at least two distinguishable values, so that we can construct (arbitrarily complex) contexts distinguishing two given terms. Hence, any positive type with at least two values would have sufficed in place of bool. For example, by applying informative boolean predicates on the naturals, distinguishable terms of type nat can be used to produce distinguishable terms of type bool.

2 Typed Equality Relation

Just as with untyped observational equivalence, the gold-standard definition is powerful, but hard to work with. We define a syntactic notion of typed equality, which should be an approximation of full typed observational equivalence.

$\boxed{\Gamma \vdash M = N : A}$

$$\frac{\Gamma \vdash M = N : A \qquad \Gamma \vdash E[\mathit{true}] = E'[\mathit{true}] : A \qquad \Gamma \vdash E[\mathit{false}] = E'[\mathit{false}] : A}{\Gamma \vdash E[M] = E'[N] : A}\ \mathsf{bool}\mathrm{X}$$

$$\frac{\Gamma,\ x : A \vdash M\ x = N\ x : B}{\Gamma \vdash M = N : A \to B}\ {\to}\mathrm{X} \qquad \frac{\Gamma \vdash M : A}{\Gamma \vdash M = M : A}\ \textsc{Refl} \qquad \frac{\Gamma \vdash M = N : A}{\Gamma \vdash N = M : A}\ \textsc{Symm}$$

$$\frac{\Gamma \vdash M = M' : A \qquad \Gamma \vdash M' = M'' : A}{\Gamma \vdash M = M'' : A}\ \textsc{Trans}$$

$$\frac{\Gamma \vdash M = N : A \qquad \Gamma' \vdash C[M] : B \qquad \Gamma' \vdash C[N] : B}{\Gamma \vdash C[M] = C[N] : B}\ \textsc{Compat}$$

$$\frac{\Gamma \vdash M : A \qquad M \mapsto_\beta M' \qquad \Gamma \vdash M' = N : A}{\Gamma \vdash M = N : A}\ \textsc{Step}$$

$\mathsf{bool}X$ and $\to X$ represent extensionality principles for boolean and function types. The rule for booleans allows us to reason by cases, and we can use it to prove the desired equations involving *not* and *and*. This expression of function extensionality essentially states that functions are equal if their bodies are equal. We also include four fundamental principles that equality should possess: reflexivity, symmetry, transitivity, and compatibility with arbitrary contexts. Finally, Step asserts that beta reductions preserve typed equality.

We can also express these extensional equations in a similar manner to beta reduction. These eta rules are interderivable with the extensionality inference rules above.

$(\eta\ \to)$	$(\lambda x.\ M\ x) =_\eta M$	$: A \to B$	(if $x \notin FV(M)$)
$(\eta\ \mathsf{bool})$	$(\mathsf{if}\ M\ \mathsf{then}\ E[\mathit{true}]\ \mathsf{else}\ E[\mathit{false}]) =_\eta E[M]$	$: A$	(if $M : \mathsf{bool}$)

3 Typed Equivalence Logical Relation

To prove that typed equality implies typed observational equivalence, we choose the logical relations approach. Our logical relation for each type will be a binary relation on terms since it corresponds to the binary relations of equality and observational equivalence.

$$\begin{array}{rll} M\ [\![\mathsf{bool}]\!]\ M' & \text{iff} & M \mapsto\!\!\!\!\!\rightarrow_\beta \mathit{true}\ {}_\beta\!\!\!\!\!\leftarrow\!\!\!\!\!\mapsto M' \text{ or } M \mapsto\!\!\!\!\!\rightarrow_\beta \mathit{false}\ {}_\beta\!\!\!\!\!\leftarrow\!\!\!\!\!\mapsto M' \\ M\ [\![A \to B]\!]\ M' & \text{iff} & \forall N, N', N\ [\![A]\!]\ N' \text{ implies } M\ N\ [\![B]\!]\ M'\ N' \end{array}$$

We will also provide an interpretation for the judgment $\Gamma \vdash M = N : A$. Since this judgment represents a proposition, its logical interpretation will be a proposition. We will choose this interpretation such that the fundamental property can be stated simply as: if $\Gamma \vdash M = N : A$ is derivable, then $[\![\Gamma \vdash M = N : A]\!]$.

$$\begin{array}{rll} \gamma\ [\![\Gamma]\!]\ \gamma' & \text{iff} & \forall x : A \in \Gamma, x[\gamma]\ [\![A]\!]\ x[\gamma'] \\ [\![\Gamma \vdash M = N : A]\!] & \text{iff} & \forall \gamma\ [\![\Gamma]\!]\ \gamma', M[\gamma]\ [\![A]\!]\ N[\gamma'] \end{array}$$

Each inference rule can also be given an interpretation, as the implication from the interpretations of its premises to the interpretation of its conclusion. Since these correspond to the inductive cases of the proof of the fundamental property, it suffices to prove the interpretations of each rule.

For each rule:

$$\left[\!\!\left[\frac{H_1 \qquad \ldots \qquad H_n}{J}\right]\!\!\right] \quad \text{iff} \quad [\![H_1]\!] \text{ and } \ldots \text{ and } [\![H_n]\!] \text{ imply } [\![J]\!]\ .$$

As for proving these cases, and thus the fundamental lemma, there is a trick to make the proof easier. The first is that the general compatibility rule COMPAT is unwieldy. Instead, each typing rule can be doubled to form a typed equality rule. As a representative example:

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B} \to\text{E} \quad \text{generates} \quad \frac{\Gamma \vdash M = M' : A \to B \qquad \Gamma \vdash N = N' : A}{\Gamma \vdash M\ N = M'\ N' : B} \to\text{E}^2$$

With these doubled rules, the REFL and COMPAT are provably unnecessary.

There is a second critical lemma for proving the fundamental property: closure of the logical relation under expansion.

Lemma 3.1 (Expansion)**.** *If* $M \mapsto_\beta M'$ $[\![A]\!]$ N' ${}_\beta\!\!\leftarrow\!\!\!\shortmid N$*, then* M $[\![A]\!]$ N*.*

Closure under expansion can be thought of as the interpretation of a new rule EXPAND (in the case of an empty environment):

$$\frac{M \mapsto_\beta M' \qquad \Gamma \vdash M = N' : A \qquad N \mapsto_\beta N'}{\Gamma \vdash M = N : A}\ \textsc{Expand}$$

If added to the system, this rule would break the property that if $\Gamma \vdash M = N : A$ is derivable, then so is $\Gamma \vdash M : A$. But since its interpretation is provable, it is sound with respect to the logical relation.

Using this lemma, along with the lemma that each logical relation is a *partial equivalence relation* (i.e. is symmetric and transitive), the fundamental property can be proven, from which it can be shown that typed equality implies typed observational equivalence.

4 Type Quantifiers

To extend our language with generic programming and modules with hidden implementations, we can add universally and existentially quantified types. Non-intuitively, exestentials enable us to define modules in which the implementation and the interface the client interacts are independent. For instance, a client of a queue module should not notice any different behavior if the implementer switches the internal queue representation to a linked list from an array. This addition of new constructs involves introducing type variables, introducing new type forms, and adding introduction and elimination term forms for both quantifiers.

$$\begin{aligned}
\textit{TypeVar} \ni\ & \alpha, \beta ::= ... \\
\textit{Types} \ni\ & A, B ::= \alpha \mid A \to B \mid \forall\alpha.\ A \mid \exists\alpha.\ A \\
\textit{Terms} \ni\ & M, N ::= x \mid \lambda x.\ \mid M\ NM \mid \Lambda\alpha.\ M \mid M\ A \mid (A,\ M) \mid \mathsf{case}\ M\ \mathsf{of}\ (\alpha,\ x.\ B) \Rightarrow N
\end{aligned}$$

The new typing rules are as follows:

$$\frac{\Theta, \alpha; \Gamma \vdash M : B}{\Theta; \Gamma \vdash \Lambda\alpha.\ M : \forall\alpha.\ B}\ \forall\text{I} \qquad \frac{\Theta; \Gamma \vdash M : \forall\alpha.\ B \qquad \Theta \vdash A : \star}{\Theta; \Gamma \vdash M\ A : B[A/\alpha]}\ \forall\text{E} \qquad \frac{\Theta; \Gamma \vdash M : B[A/\alpha] \qquad \Theta \vdash A : \star}{\Theta; \Gamma \vdash (A,\ M) : \exists\alpha.\ B}\ \exists\text{I}$$

$$\frac{\Theta; \Gamma \vdash M : \exists\alpha.\ B \qquad \Theta, \alpha; \Gamma, x : B \vdash N : C \qquad \alpha \notin \mathrm{FV}(C)}{\Gamma \vdash \mathsf{case}\ M\ \mathsf{of}\ (\alpha,\ x.\ B) \Rightarrow N : \mathsf{bool}}\ \exists\text{E}$$

Here, $\Theta \vdash A : \star$ ensures that the variables in A are either bound or appear in Θ. Note the last premise of $\exists$E which ensures that C does not leak the type variable α outside of its scope.

The new rules of our operational semantics are:

$$\begin{aligned}
&(\beta\forall)(\Lambda\alpha.M)A \mapsto_\beta M[A/\alpha] \\
&(\beta\exists)\text{Case (A, M) of } (\alpha, x : B) \Rightarrow N \mapsto_\beta N[A/\alpha, M/x]
\end{aligned}$$

4.1 Extending the Logical Relation

When we try to extend the logical relation, we are stuck at the definition of $[\![\alpha]\!]$. We have no knowledge of what α is, until we revise the logical relation to take a type environment τ, a partial map from type variables to semantic types $[\![A]\!]$, as an additional parameter. Then we have:

$$M\ [\![\alpha]\!]_\tau\ M' \quad \text{iff} \quad M\ \tau(\alpha)\ M'.$$

But then a naive definition for the interpretation of quantifiers might look something like this:

$$M\ [\![\forall\alpha.\ B]\!]_\tau\ M' \quad \text{iff} \quad \forall A \in \mathit{Type}.\ M\ A\ [\![B]\!]_{\tau[[\![A]\!]/\alpha]}\ M'\ A$$

$$M\ [\![\exists\alpha.\ B]\!]_\tau\ M' \quad \text{iff} \quad M \mapsto\!\!\!\!\rightarrow_\beta (A, N) \wedge M' \mapsto\!\!\!\!\rightarrow_\beta (A', N') \wedge N\ [\![B]\!]_{\tau,[\![A?]\!]_\tau/\alpha}\ N'.$$

Here we run into trouble, since the meaning of $\forall\alpha.\ B$ depends on $[\![A]\!]$ for an arbitrary (potentially larger) A, jeopardizing the well-foundedness of our relation. In addition, this defines a deceptive interpretation of the existentials, where two different implementations should be related. This could be fixed using techniques like step-indexing (as seen in Amal's lectures), but in fact, we don't need to go that far in our case.

Instead, we can generalize our approach by including an extra parameter $C \subseteq \mathit{Rel}(\mathit{Term}, \mathit{Term})$ to our relation. This relation C represents any relation candidate $\in RC$. A relation candidate is any binary relation on terms that is closed under expansion. In other words, a relation $\mathbb{A}$ is a relation candidate if for terms M and N, M reduces to M' in many beta steps and N reduces to N' in many beta steps and M' $\mathbb{A}$ N' then M $\mathbb{A}$ N. An equivalence candidate $\in EC$ is a relation candidate that is an equivalence relation, a relation that is reflexive, symmetric, and transitive. The types inside a relation candidate may not be in harmony with the types in our language, those types might be related to types in our language but they are not required to. This abstraction enables us to overcome impredicativity and to update our definitions as follows:

$$M\ [\![\forall\alpha.\ B]\!]^C_\tau\ M' \quad \text{iff} \quad \forall A, A' \in \mathit{Type}, \mathbb{A} \in C.\ M\ A\ [\![B]\!]^C_{\tau[\mathbb{A}/\alpha]}\ M'\ A'.$$

Now the relation is well-founded, but the definition itself seems dubious. We have no requirements on the relationship on A, A', and C! However, due to the magic of parametricity, this turns out to work. Our new rule for existential quantification similarly becomes:

$$M\ [\![\exists\alpha.\ B]\!]^C_\tau\ M' \quad \text{iff} \quad M \mapsto\!\!\!\!\rightarrow_\beta (A, N) \text{ and } M \mapsto\!\!\!\!\rightarrow_\beta (A', N') \text{ and } \exists\mathbb{A} \in C.\ N\ [\![B]\!]^C_{\tau[\mathbb{A}/\alpha]}$$