

Semantics of Functional Programming

Lecture IV: Computational Adequacy

Chen, Liang-Ting
lxc@iis.sinica.edu.tw

Formosan Summer School on Logic, Language, and Computation 2014

Overview

So far we have given two kinds of semantics for **PCF**. For a program M of type σ ,

- one gives how the program M is evaluated to a value V via the reduction relation $M \Downarrow V$;
- the other defines what the denotation $\llbracket M \rrbracket$ of M is in a domain D_σ .

In this lecture, we will compare these two approaches and discuss some issues arising from them:

Correctness $M \Downarrow V$ implies $\llbracket M \rrbracket = \llbracket V \rrbracket$.

Completeness $\llbracket M \rrbracket = n$ implies $M \Downarrow n$

Computational adequacy Both of correctness and completeness hold.

1 Correctness

nat values always converges

The bottom element \perp models the divergence of computation. A value of **nat** is meant to be some natural number, so it shouldn't diverge.

Lemma 1. *For every value V of type **nat**, the denotation $\llbracket V \rrbracket$ is an element of \mathbb{N} . In particular, $\llbracket V \rrbracket \neq \perp$.*

Proof. By structural induction on values:

$$\begin{array}{c} \hline \text{zero val} \\ \hline \text{M val} \\ \text{suc M val} \\ \hline \text{M term} \\ \lambda x. M \text{ val} \end{array}$$

□

Theorem 2. *For every two programs M and V , $M \Downarrow V$ implies $\llbracket M \rrbracket = \llbracket V \rrbracket$.*

Proof sketch. Prove $\llbracket M \rrbracket = \llbracket V \rrbracket \in \llbracket \tau \rrbracket$ by structural induction on the derivation of $M \Downarrow V$. □

We show the case $(\Downarrow\text{-suc})$ first and the cases $(\Downarrow\text{-zero})$ and $(\Downarrow\text{-lam})$ are similar and easy.

- For $(\Downarrow\text{-suc})$, we show that $\llbracket \text{suc } M \rrbracket = \llbracket \text{suc } V \rrbracket$ if $\llbracket M \rrbracket = \llbracket V \rrbracket$. By definition, we simply calculate its denotation directly:

$$\llbracket \text{suc } M \rrbracket = S(\llbracket M \rrbracket) = S(\llbracket V \rrbracket) = \llbracket \text{suc } V \rrbracket$$

where the middle equality follows from the induction hypothesis.

Try to do the cases $(\Downarrow\text{-zero})$, $(\Downarrow\text{-lam})$, and $(\Downarrow\text{-ifz}_0)$.

The case $(\Downarrow\text{-app})$ is slightly complicated, as we have to address the binding structure using Substitution Lemma.

- For $(\Downarrow\text{-app})$, we show that $\llbracket M N \rrbracket = \llbracket V \rrbracket$ if $\llbracket M \rrbracket = \llbracket \lambda x. E \rrbracket$ and $\llbracket E[N/x] \rrbracket = \llbracket V \rrbracket$. We calculate the denotation as follows

$$\begin{aligned} \llbracket M N \rrbracket &= ev(\llbracket M \rrbracket, \llbracket N \rrbracket) \\ &= ev(\llbracket \lambda x. E \rrbracket, \llbracket N \rrbracket) \\ &= ev(\llbracket x : \sigma \vdash E : \tau \rrbracket, \llbracket N \rrbracket) \\ &= \llbracket x : \sigma \vdash E : \tau \rrbracket(\llbracket N \rrbracket) = \llbracket E[N/x] \rrbracket = \llbracket V \rrbracket \end{aligned}$$

where the last but one equation follows from Substitution Lemma.

- Complete the remaining two (interesting) cases $(\Downarrow\text{-ifz}_1)$ and $(\Downarrow\text{-fix})$. *Hint.* Consider Substitution Lemma and the properties of the fixpoint operator μ .

2 Equational reasoning

Logical equivalence

Definition 3 (Applicative approximation). For each type σ , define a relation \preceq_σ between programs of σ :

1. For the type **nat**, define

$$M \lesssim_{\text{nat}} N$$

if for all $n \in \mathbb{N}$, $M \Downarrow \underline{n}$ implies $N \Downarrow \underline{n}$

2. For every type $\sigma \rightarrow \tau$, define

$$M \lesssim_{\sigma \rightarrow \tau} N$$

if for all program P , $M P \lesssim_{\tau} N P$.

Two programs M and N of the same type σ are **logically equivalent** denoted $M \simeq_{\sigma} N$ if $M \lesssim_{\sigma} N$ and $M \gtrsim_{\sigma} N$.

The relation \lesssim_{σ} is a preorder, so \simeq_{σ} is indeed an equivalence.

Proposition 4. *The logical equivalence \simeq_{σ} is an equivalence relation.*

A program M can be replaced by another program N without changing results if $M \simeq_{\sigma} N$.

Example 5. The following two programs are logically equivalent:

$$\lambda x. x \quad \text{and} \quad \lambda x. \text{pred}(\text{suc } x)$$

Reduction respects logical equivalence

Recall that from $M \rightsquigarrow^* M'$ and $M' \Downarrow V$ it follows that $M \Downarrow V$ in the agreement between \rightsquigarrow and \Downarrow .

Proposition 6. *Let M and M' be programs of type σ . If $M \rightsquigarrow^* M'$, then $M \lesssim_{\sigma} M'$.*

The other direction follows from the determinacy and values cannot be reduced further:

Proposition 7. *For every $M \Downarrow V$ and $M \rightsquigarrow^* M'$, we have $M' \Downarrow V$.*

Corollary 8 (Reduction entails logical equivalence). *If $M \rightsquigarrow^* M'$, then $M \simeq_{\sigma} M'$.*

However, logical equivalence goes beyond reduction. Consider the following two programs of type **nat** \rightarrow **nat** \rightarrow **nat**:

$$\lambda x. \lambda y. x + y$$

and

$$\lambda x. \lambda y. y + x$$

Surely the addition of natural numbers are commutative, but *why*? By definition they are already values, so they cannot be reduced to each other.

Remark 2.1. We can show directly that these two programs are logically equivalent in dependent type theory. Yet, we will present an external approach using denotational semantics in the absense of the identity type.

3 Computational Adequacy

In the following, we will show that for every program M of type **nat** if $\llbracket M \rrbracket = n$ then M reduces to the numeral \underline{n} .

- Define a relation R_{σ} for each type σ between the domain $\llbracket \sigma \rrbracket = D_{\sigma}$ and the collection of programs of type σ :

$$R_{\sigma} \subseteq D_{\sigma} \times \text{Prg}_{\sigma}$$

for every type σ where $\text{Prg}_{\sigma} = \{ M \mid \vdash M : \sigma \}$.

- Then prove that $\llbracket M \rrbracket R_{\sigma} M$ for every program M of type σ , and, by construction $\llbracket M \rrbracket R_{\text{nat}} M$ is equivalent to that $\llbracket M \rrbracket = n$ implies $M \Downarrow \underline{n}$.

With this property, we can conclude that denotational equivalence entails logical equivalence.¹

Logical relation between semantics and syntax

Definition 9 (Logical relation). For every type σ , define a relation $R_{\sigma} \subseteq D_{\sigma} \times \text{Prg}_{\sigma}$ inductively as follows:

- $d R_{\text{nat}} M$ if M reduces to \underline{n} whenever d is a natural number:

$$d R_{\text{nat}} M \quad \text{if} \quad \forall n \in \mathbb{N}. d = n \implies M \Downarrow \underline{n}$$

- for every function type, $f R_{\sigma \rightarrow \tau} M$ if the outcome is always related whenever the input is related:

$$f R_{\sigma \rightarrow \tau} M \quad \text{if}$$

$$\forall d, N. d R_{\sigma} N \implies f(d) R_{\tau} M N$$

For example, $0 R_{\text{nat}} \text{zero}$, and $n + 1 R_{\text{nat}} \text{suc } M$ wherever $n R_{\text{nat}} M$ for $n \in \mathbb{N}$.

Properties of R_{σ}

Lemma 10. *For every type σ , the following statements are true:*

1. If $d' \sqsubseteq d$ and $d R_{\sigma} M$, then $d' R_{\sigma} M$;
2. For every $M \in \text{Prg}_{\sigma}$, the set

$$R_{\sigma} M := \{ d \in D_{\sigma} \mid d R_{\sigma} M \}$$

contains \perp and is closed under directed sups;²

3. If $d R_{\sigma} M$ and $M \lesssim_{\sigma} M'$, then $d R_{\sigma} M'$.

¹ But, the converse may fail.

² Let S be an arbitrary directed subset of D_{σ} , if $d R_{\sigma} M$ for every $d \in S$, then $\bigsqcup S R_{\sigma} M$.

Proof. By induction on σ . \square

Lemma 11 (General recursion). *If we have $f R_{\sigma \rightarrow \sigma} (\lambda x. M)$, then $\mu(f) R_{\sigma} (Yx. M)$.*

Proof sketch. By definition $\mu(f)$ is the directed supremum of the following directed sequence

$$\perp \sqsubseteq f(\perp) \sqsubseteq f^2(\perp) \sqsubseteq \dots \sqsubseteq f^i(\perp) \sqsubseteq \dots,$$

so it suffices to show that

$$f^i(\perp) R_{\sigma} (Yx. M)$$

for every $i \in \mathbb{N}$, because $R_{\sigma}(Yx. M)$ is closed under directed sups. We prove it by induction on i and properties of R_{σ} . \square

The complete proof is listed below.

For $i = 0$: By definition $f^0(\perp) = \perp$, so $\perp R_{\sigma} (Yx. M)$ follows.

For $i = n + 1$: By the assumption $f R_{\sigma \rightarrow \sigma} (\lambda x. M)$, it follows that

$$f^{n+1}(\perp) R_{\sigma} (\lambda x. M) (Yx. M)$$

by the induction hypothesis $f^n(\perp) R_{\sigma} (Yx. M)$.

The RHS reduces to $M[Yx. M/x]$ and $Yx. M \rightsquigarrow M[Yx. M/x]$, so the RHS is logically equivalent to $Yx. M$. Hence, it follows that

$$f^{n+1}(\perp) R_{\sigma} (Yx. M).$$

Therefore, it follows that $\bigsqcup_{i \in \mathbb{N}} f^i(\perp) R_{\sigma} (Yx. M)$.

Substitution Lemm and completeness

Lemma 12 (Substitution). *Let $\Gamma = x_1 : \sigma_1, \dots, x_k : \sigma_k$ be a context and $d_i R_{\sigma_i} N_i$ for $i = 1, \dots, n$. For every well-typed term M we have*

$$\llbracket \Gamma \vdash M : \tau \rrbracket(\vec{d}) R_{\tau} M[\vec{N}/\vec{x}]$$

Theorem 13 (Completeness). *For every $\vdash M : \text{nat}$, we have $M \Downarrow \underline{n}$ if $\llbracket M \rrbracket = n$.*

Proof. A special case of the previous lemma:

$$\llbracket \vdash M : \tau \rrbracket(*) R_{\sigma} M$$

where the LHS is $\llbracket M \rrbracket$.

Proof of Substitution Lemma

To prove the lemma, do induction on the typing rules for **PCF**. For convenience, we write

$$\vec{d} R \vec{N} \quad \text{for} \quad d_i R_{\sigma_i} N_i \quad \text{indexed by } i = 1, \dots, n$$

where \vec{d} stands for (d_1, \dots, d_n) and \vec{N} stands for (N_1, \dots, N_n) .

(z), (s) These two cases follow from $0 R_{\text{nat}} \text{zero}$ and $n + 1 R_{\text{nat}} \text{suc } M$ whenever $n R_{\text{nat}} M$.

(var) To show that

$$\llbracket \dots, x_i : \sigma_i, \dots \vdash x_i : \sigma_i \rrbracket R_{\sigma_i} x_i[\vec{N}/\vec{x}]$$

we check both sides separately. By definition, we have

$$\llbracket \dots, x_i : \sigma_i, \dots \vdash x_i : \sigma_i \rrbracket(\vec{d}) = d_i \quad \text{and} \quad [\vec{N}/\vec{x}] = N_i.$$

Therefore, from the assumption it follows that $d_i R_{\sigma_i} N_i$ for every i .

(abs) We need to show that

$$\llbracket \Gamma \vdash \lambda x. M : \tau \rrbracket(\vec{d}) R_{\sigma \rightarrow \tau} (\lambda x. M)[\vec{N}/\vec{x}] \quad (1)$$

under the induction hypothesis

$$\llbracket \Gamma, x : \sigma \vdash M : \tau \rrbracket(\vec{d}, d) R_{\tau} M[\vec{N}, N / \vec{x}, x].$$

• For the LHS, we have by definition

$$\begin{aligned} & \llbracket \Gamma \vdash \lambda x. M : \tau \rrbracket(\vec{d})(d) \\ &= \llbracket \Gamma, x : \sigma \vdash M : \tau \rrbracket(\vec{d}, d). \end{aligned}$$

• For the RHS, we have

$$\begin{aligned} & (\lambda x. M)[\vec{N}/\vec{x}] N \\ & \rightsquigarrow (\lambda x. M)[\vec{N}/\vec{x}][N/x] \\ &= (\lambda x. M)[\vec{N}, N / \vec{x}, x] \end{aligned}$$

and it follows that these two terms are logically equivalent. Thus, (1) follows by the definition of $R_{\sigma \rightarrow \tau}$.

(Y) We show that $\llbracket \Gamma \vdash Yx. M : \sigma \rrbracket(\vec{d}) R_{\sigma} (Yx. M)[\vec{N}/\vec{x}]$ under the assumption that

$$\llbracket \Gamma, x : \sigma \vdash M : \sigma \rrbracket(\vec{d}, d) R_{\sigma} M[\vec{N}, N / \vec{x}, x] \quad (2)$$

Recall the lemma for general recursion. It suffices to show $\Lambda \llbracket \Gamma, x : \sigma \vdash M : \sigma \rrbracket(\vec{d}) R_{\sigma \rightarrow \sigma} \lambda x. M[\vec{N}/\vec{x}]$ or, equivalently

$$\llbracket \Gamma, x : \sigma \vdash M : \sigma \rrbracket(\vec{d}, d) R_{\sigma} (\lambda x. M[\vec{N}/\vec{x}]) N \quad (3)$$

for every $d R_{\sigma} N$. The RHS can be reduced to

$$M[\vec{N}/\vec{x}][N/x] = M[\vec{N}, N / \vec{x}, x],$$

so (2) implies (3) by logical equivalence.

\square **(app), (ifz)** Exercises.

3.1 Applications of adequacy

Applicative approximation coincides with logical relation

Lemma 14. *For every $\vdash M : \sigma$ and $\vdash N : \sigma$,*

$$M \lesssim_\sigma N \quad \text{if and only if} \quad \llbracket M \rrbracket R_\sigma N.$$

Proof. $M \lesssim_\sigma N$. By adequacy, we have $\llbracket M \rrbracket R_\sigma M$, so $\llbracket M \rrbracket R_\sigma N$.

$\llbracket M \rrbracket R_\sigma N$. Prove it by induction on σ .

nat: If $\llbracket M \rrbracket R_{\text{nat}} N$, then $N \Downarrow \underline{n}$ whenever $\llbracket M \rrbracket = n$.

$\sigma \rightarrow \tau$: For $\sigma \rightarrow \tau$, by adequacy, we have $\llbracket P \rrbracket R_\sigma P$ for every P , so by assumption and $\llbracket M P \rrbracket = \llbracket M \rrbracket(\llbracket P \rrbracket) R_\tau N P$. By induction hypothesis, $M P \lesssim_\tau N P$ for every P , so $M \lesssim_{\sigma \rightarrow \tau} N$ by definition. \square

Corollary 15. *Given two $\vdash M : \sigma$ and $\vdash N : \sigma$, if $\llbracket M \rrbracket = \llbracket N \rrbracket$, then M and N are logically equivalent.*

Proof. 1. By adequacy $\llbracket M \rrbracket R M$ and by assumption $\llbracket N \rrbracket = \llbracket M \rrbracket R M$, it follows that $N \lesssim M$.

2. Similarly, $\llbracket M \rrbracket R N$, so $M \lesssim N$.

Hence, M and N are logically equivalent. \square

From this property, techniques and results in denotational semantics can be used to argue logical equivalence and reductions.

Compactness

Recall that the semantics of general recursion is the least upper bound of its finite unfoldings

$$\llbracket Yx.M \rrbracket = \bigsqcup_{i \in \mathbb{N}} \llbracket Y^i x.M \rrbracket$$

where $Y^i x.M$ is defined inductively by

1. $Y^0 x.M := Yx.x$ and
2. $Y^{n+1} x.M := M[Y^n x.M/x]$

and $\llbracket Y^i x.M \rrbracket = \llbracket \lambda x.M \rrbracket^i(\perp)$.

Theorem 16. *Suppose that $x \neq y$,*

$$y : \sigma \vdash E : \text{nat} \quad \text{and} \quad \vdash Yx.M : \sigma.$$

If $E[Yx.M/y] \Downarrow \underline{n}$ then $E[Y^m x.M/y] \Downarrow \underline{n}$ for some m .

Proof. By the Substitution Lemma, we have

$$\llbracket E[Yx.M/y] \rrbracket = \llbracket y : \sigma \vdash E : \text{nat} \rrbracket(\llbracket Yx.M \rrbracket).$$

Let $g := \llbracket y : \sigma \vdash E : \text{nat} \rrbracket$ and $f := \llbracket x : \sigma \vdash M : \sigma \rrbracket$.

$$\begin{aligned} \llbracket y : \sigma \vdash E : \text{nat} \rrbracket(\llbracket Yx.M \rrbracket) &= g(\mu f) \\ &= g\left(\bigsqcup_{i \in \mathbb{N}} f^i(\perp)\right) \\ &= \bigsqcup_{i \in \mathbb{N}} (g \circ f^i)(\perp) = n \end{aligned}$$

Therefore there exists some $m \in \mathbb{N}$ such that $(g \circ f^m)(\perp) = n$. By adequacy, it follows that $E[Y^m x.M/y] \Downarrow \underline{n}$. \square

Finite unfoldings approximate general recursion

Lemma 17. *Suppose that $x : \sigma \vdash M : \sigma$. Then for every $i \in \mathbb{N}$, we have*

$$Y^i x.M \lesssim_\sigma Yx.M.$$

The proof is left as an exercise.

Theorem 18 (Fixed Point Induction). *Suppose that $x : \sigma \vdash M : \sigma$, $x : \sigma \vdash N : \sigma$ and*

$$Y^i x.M \simeq_\sigma Y^i x.N$$

for every $i \in \mathbb{N}$. Then, we also have

$$Yx.M \simeq_\sigma Yx.N$$

Proof. We show that $Yx.M \lesssim_\sigma Yx.N$, or equivalently $\llbracket Yx.M \rrbracket R_\sigma \llbracket Yx.N \rrbracket$, and the other direction follows similarly.

Let $f := \llbracket x : \sigma \vdash M : \sigma \rrbracket$ and $g := \llbracket x : \sigma \vdash N : \sigma \rrbracket$. Since the set

$$R_\sigma(Yx.N) = \{d \in D_\sigma \mid d R_\sigma Yx.N\}$$

is closed under directed supremum, it suffices to show that

$$\llbracket Y^i x.M \rrbracket R_\sigma Yx.N$$

for every i .

By assumption, we have $\llbracket Y^i x.M \rrbracket R_\sigma Y^i x.N$, so it suffices to show that $Y^i x.N \lesssim_\sigma Yx.N$. By the previous lemma the statement follows. \square

Exercises

Show that the following pairs of programs are logically equivalent.

1.