

Comp 311
Principles of Programming Languages
Lecture 11
The Semantics of Recursion II

Corky Cartwright
September 18, 2009

Recursive Definitions

- Given a Scott-domain D , we can write equations of the form:

$$\mathbf{f} = \mathbf{E}_f$$

where \mathbf{E}_f is an expression constructed from constants in D , operations (continuous functions) on D , and \mathbf{f} .

- Example: let D be the domain of Scheme values. Then

fact =

(lambda (n) (if (zero? n) 1 (* n (fact (- n 1)))))

is such an equation.

- Such equations are called *recursive definitions*.

Solutions to Recursion Equations

Given an equation:

$$\mathbf{f} = \mathbf{E}_f$$

what is a solution? All of the constants and operations in \mathbf{E}_f are known except \mathbf{f} .

A solution is any function \mathbf{f}^* such that

$$\mathbf{f}^* = \mathbf{E}_{f^*}$$

is a solution. But there may be more than one solution. We want to select the “best” solution. Note that \mathbf{f}^* is an element of whatever domain D^* is the type of \mathbf{E}_f . In the most common case, it is $D \rightarrow D$, but it can be D , $D^k \rightarrow D$, ... The best solution (the one that always exists, is unique, and is *computable* is the *least* solution under the approximation ordering in D^*).

Constructing the Least Solution

How do we know that any solution exists to the equation $\mathbf{f} = \mathbf{E}_f$?

We will construct the least solution and prove it is a solution!

Since the domain D^* for \mathbf{f} is a Scott-Domain, it has a least element \mathbf{bot}_{D^*} . Hence, \mathbf{bot}_{D^*} approximates every solution to the equation

$$\mathbf{f} = \mathbf{E}_f.$$

Now form the function $\mathbf{F}: D^* \rightarrow D^*$ defined by

$$\mathbf{F}(\mathbf{f}) = \mathbf{E}_f$$

or equivalently,

$$\mathbf{F} = \lambda \mathbf{f} . \mathbf{E}_f$$

Consider the sequence $\mathbf{S}: \mathbf{bot}_{D^*}, \mathbf{F}(\mathbf{bot}_{D^*}), \mathbf{F}(\mathbf{F}(\mathbf{bot}_{D^*})), \dots, \mathbf{F}^k(\mathbf{bot}_{D^*}), \dots$

Claim: \mathbf{S} is an ascending chain (chain for short) in $D^* \rightarrow D^*$.

Proof. $\mathbf{bot}_{D^*} \leq \mathbf{F}(\mathbf{bot}_{D^*})$ by the definition of \mathbf{Bot}_{D^*} . If $\mathbf{M} \leq \mathbf{N}$, then $\mathbf{F}(\mathbf{M}) \leq \mathbf{F}(\mathbf{N})$ by monotonicity. Hence, $\mathbf{F}^k(\mathbf{bot}_{D^*}) \leq \mathbf{F}^{k+1}(\mathbf{bot}_{D^*})$ for all k . Q.E.D.

Claim: \mathbf{S} has a least upper bound \mathbf{f}^*

Proof. Trivial. \mathbf{S} is a chain in D^* and hence must have a least upper bound because D^* is a Scott-Domain.

Proving \mathbf{f}^* is a fixed point of \mathbf{F}

Must show: $\mathbf{F}(\mathbf{f}^*) = \mathbf{f}^*$ where $\mathbf{F} = \lambda \mathbf{f} . \mathbf{E}_f$.

Claim: By definition $\mathbf{f}^* = \cup \mathbf{F}^k(\mathbf{bot}_{D^*})$. Since \mathbf{F} is continuous

$$\begin{aligned}\mathbf{F}(\mathbf{f}^*) &= \mathbf{F}(\cup \mathbf{F}^k(\mathbf{bot}_{D^*})) \\ &= \cup \mathbf{F}^{k+1}(\mathbf{bot}_{D^*}) && \text{(by continuity)} \\ &= \cup \mathbf{F}^k(\mathbf{bot}_{D^*}) && \text{(since } \mathbf{bot}_{D^*} \leq \mathbf{F}(\mathbf{bot}_{D^*}) \text{)} \\ &= \mathbf{f}^*\end{aligned}$$

Q.E.D.

Note: all of the steps in the preceding proof are trivial except for the step justified by continuity.

Examples

Look at factorial in detail using DrScheme.

How Can We Compute \mathbf{f}^* Given \mathbf{F} ?

Need to construct $\mathbf{F}^\infty(\perp)$ from \mathbf{F} . Let

$$\mathbf{Y}(\mathbf{F}) = \mathbf{f}^* = \mathbf{F}^\infty(\perp).$$

Can we write code for \mathbf{Y} ?

Idea: use syntactic trick in Ω to build a potentially infinite stack of \mathbf{F} s.

- Preliminary attempt:

$$(\lambda \mathbf{x}. \mathbf{F}(\mathbf{x} \ \mathbf{x})) \ (\lambda \mathbf{x}. \mathbf{F}(\mathbf{x} \ \mathbf{x}))$$

- Reduces to (in one step):

$$\mathbf{F} \ ((\lambda \mathbf{x}. \mathbf{F}(\mathbf{x} \ \mathbf{x})) \ (\lambda \mathbf{x}. \mathbf{F}(\mathbf{x} \ \mathbf{x})))$$

- Reduces to (in k steps):

$$\mathbf{F}^k \ ((\lambda \mathbf{x}. \mathbf{F}(\mathbf{x} \ \mathbf{x})) \ (\lambda \mathbf{x}. \mathbf{F}(\mathbf{x} \ \mathbf{x})))$$

What Is the Code for **Y**?

$\lambda F. (\lambda x. F(x x)) (\lambda x. F(x x))$

- Does this work for Scheme (or Java with an appropriate encoding of functions as anonymous inner classes)? No!
- Why not? What about divergence? Assume **G** is a λ -expression defining a functional like FACT

$$\begin{aligned} & (\lambda F. (\lambda x. F(x x)) (\lambda x. F(x x))) G \\ &= G((\lambda x. G(x x)) (\lambda x. G(x x))) \\ &= \dots \text{ (diverging)} \end{aligned}$$

What If We Use Call-by-name?

By assumption \mathbf{G} must have the form $(\lambda \mathbf{f}. (\lambda \mathbf{n}. \mathbf{M}))$

$$\begin{aligned} & (\lambda \mathbf{F}. (\lambda \mathbf{x}. \mathbf{F}(\mathbf{x} \ \mathbf{x})) (\lambda \mathbf{x}. \mathbf{F}(\mathbf{x} \ \mathbf{x}))) \ \mathbf{G} \\ = & \ \mathbf{G} \ ((\lambda \mathbf{x}. \ \mathbf{G}(\mathbf{x} \ \mathbf{x})) (\lambda \mathbf{x}. \ \mathbf{G}(\mathbf{x} \ \mathbf{x}))) \\ = & \ (\lambda \mathbf{f}. (\lambda \mathbf{n}. \ \mathbf{M})) \ ((\lambda \mathbf{x}. \ \mathbf{G}(\mathbf{x} \ \mathbf{x})) (\lambda \mathbf{x}. \ \mathbf{G}(\mathbf{x} \ \mathbf{x}))) \\ = & \ (\lambda \mathbf{n}. \ \mathbf{M}[\mathbf{x} := (\lambda \mathbf{x}. \ \mathbf{G}(\mathbf{x} \ \mathbf{x})) (\lambda \mathbf{x}. \ \mathbf{G}(\mathbf{x} \ \mathbf{x}))]) \end{aligned}$$

If the evaluation \mathbf{M} of does not require evaluating an occurrence of \mathbf{f} , then \mathbf{x} is not evaluated. Otherwise, the binding of \mathbf{x} is unwound only as many times as required to get to the base case in the definition $\mathbf{f} = \lambda \mathbf{n}. \mathbf{M}$.

Exercise: how can we workaroud this problem to create a version of the \mathbf{Y} operator that works for call-by-value Scheme and Jam? Hint: if \mathbf{M} is a divergent term denoting a unary function $\lambda \mathbf{x}. \mathbf{M}\mathbf{x}$ is an equivalent term that is not divergent! (As a concrete example, assume that \mathbf{M} is the term Ω .)