Intro to Denotational Semantics

Eric L. McCorkle

December 13, 2015

Semantics

Semantics seeks to define meaning of a programming language.

- Informal prose is a bad representation: mathematics provides better tools.
- Two main methods: *Operational* and *Denotational*.
- We will focus on denotational methods here.

Operational Methods, Briefly

Operational methods focus on *term rewriting systems*.

- Rewrite rules of the form term $\rightarrow$ term
- Example: `if true` t_t `else` $t_f \rightarrow t_t$
- Many variations: small-step, big-step, contextual, etc.
- Type systems need safety proofs (all well-typed terms can execute, execution of well-typed terms yields well-typed terms).

Denotational Methods

Denotational methods define the meaning of a language by mapping each term to a mathematical object.

- Denotational methods can directly describe infinite and infinite-rank objects without much trouble
- Multiple "dimensions" of induction are much easier to accomplish
- Easier to analyze the language mathematically

Denotational Methods: Approaches

General approach:

- Define a structured space, or *domain* which contains all possible values in the language.
- Model datatype constructors using *constructions* on spaces (ex. cartesian products, disjoint unions, function spaces)
- Handle recursion, infinite/infinite-rank objects with *limits*.

Two main foundations:

- Complete partial orders (domain theory)
- Metric spaces

Domain Theory (Scott-Strachey Semantics)

Scott-Strachey semantics, also known as *domain theory* was the first approach to denotational semantics.

- Spaces of values form directed-complete partial orders
- Order is an order of "definiteness"
- Refinement of definitions is a continuous monotone function
- Infinite objects are fixed-points of refinement

Partial Orders

A *partial order* is essentially a total order where some elements cannot be compared.
We write the order relation as $\sqsubseteq$ instead of $\leq$. It has the following properties:

- $x \sqsubseteq x$
- if $x \sqsubseteq y$ and $y \sqsubseteq z$ then $x \sqsubset z$
- if $x \sqsubseteq y$ and $y \sqsubseteq x$ then $x = y$

The strict form $\sqsubset$ is often called "below". Sometimes $\sqsubseteq$ is misleadingly called "below". Some authors use $\sqsubset$ and $\not\sqsupseteq$.

Meets and Joins

- The *join* (or *least upper bound*, or *supremum*) $\bigsqcup_i a_i$ of elements a_i is an element a such that $a_i \sqsubseteq a$ for all i and there is no $a' \sqsubset a$ with this property.
- The *meet* (or *greatest lower bound*, or *infimum*) $\bigsqcap_i a_i$ of elements a_i is an element a such that $a \sqsubseteq a_i$ for all i and there is no $a \sqsubset a'$ with this property.
- A *join semilattice* is a partial order in which every two elements have a join.
- A *meet semilattice* is a partial order in which every two elements have a meet.
- A *directed subset* of a partial order is a subset that is a join-semilattice.

Chains, Continuity, and Fixed-Points

- A *chain* is a sequence of elements a_i such that $a_1 \sqsubseteq a_2 \sqsubseteq \ldots$.
- A *monotone* function f has the property that $a \sqsubseteq f(a)$ (note that monotone functions define a chain.
- A *(Scott-)continuous* function f has the property that $f(\bigsqcup_i a_i) = \bigsqcup_i f(a_i)$
- A *fixed point* of a function f is a point a such that $f(a) = a$
- A *chain-complete* partial order (CCPO) is a partial order in which all ω-chains have a join.
- A *directed-complete* partial order (DCPO) is a partial order in which all directed subsets have a join.

An important result is that CCPO and DCPO are equivalent.

Order of Definiteness

Domain theory is based on DCPO's, where the order relation describes an *order of definiteness* or *order of approximation*.

- $\perp$ represents an "indeterminate" value.
- Non-terminating computations are represented by $\perp$.
- If $a \subseteq b$, then a is "less definite" than b.
- Example: a function $f(x) = \{1 \mapsto 1, 2 \mapsto 2\}$ is more definite than $f'(x) = \{1 \mapsto 1\}$, thus $f' \sqsubset f$.
- Continuous monotone function "refine" terms by adding more definition.

Example: Factorial

The factorial function is a good example. We start with a base definition:

$$!_0 = \{0 \mapsto 1, 1 \mapsto \bot, \ldots\}$$

We get a progressively more defined version by "unfolding":

$$!_1 = \{0 \mapsto 1, 1 \mapsto 1, 2 \mapsto \bot, \ldots\} \sqsubset$$

$$!_2 = \{0 \mapsto 1, 1 \mapsto 1, 2 \mapsto 2, 3 \mapsto \bot, \ldots\} \sqsubset$$

$$!_3 = \{0 \mapsto 1, 1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 6, 4 \mapsto \bot, \ldots\}$$

and so on. The upper-bound of this chain is !.

Constructions

We represent datatype constructors using *constructions* on DCPO's. Each of these is again a DCPO:

- *Cartesian products* $A \times B$ and *smash products* $A \otimes B$ represent lazy and strict tuples.
- *Continuous function spaces* $[A \to B]$ and *strict* function spaces $[A \to_! B]$ represent lazy and strict functions.
- *Disjoint* or *labeled* unions $A \cup B$ (often $A \oplus B$) "glue" spaces together.
- The *pointed* construction $A_\perp$ adds a $\perp$ element.

We can build all the "classic" compound datatypes using these constructions.

Embedding/Projections and Retracts

In general, for some construction on DCPO's $F(A_1, A_2, \ldots)$, where A, B are DCPO's, we have pairs of functions (e_i, p_i) called *embedding/projection pairs* with the following properties:

- e_i (called the *embedding*) maps each $a \in A_i$ to a unique element of $F(A_1, A_2, \ldots)$ (injective)
- p_i (called the *projection*) maps a unique $a \in F(A_1, A_2, \ldots)$ to each element of A_i (surjective)
- e_i and p_i are inverses in A_i ($p_i \circ e_i = \mathrm{id}_{A_i}$), but not necessarily in $F(A_1, A_2, \ldots)$

Some space A is a *retract* of B if there is an embedding/projection pair that preserves the order properties.

Facts about Embedding/Projections and Retracts

The following are true of embedding/projection pairs and retracts:

- Embedding/projection pairs compose nicely. If (e_1, p_1) goes from A to B and (e_2, p_2) goes from B to C are, $(e_2 \circ e_1, p_1 \circ p_2)$ goes from A to C.
- Retracts are transitive. If A is a retract of B and B is a retract of C, then A is also a retract of C (get the embedding/projection pair as desrcibed above).
- A space can be a rectract of itself, in which case the embedding/porjection pair is a true isomorphism.

Recursive Domain Equations

Domains with only finite objects can be built using methods developed so far. Domains with *infinite* objects (infinite-rank types, higher-order functions, etc) can be characterized with recursive domain equations.
For example, the following recursive domain equation characterizes λ-calculus:

$$D \cong [D \to D]$$

A more general construction might look like this:

$$D \cong D_0 \cup (D \times D) \cup [D \to D]$$

where D_0 is a *basis set* consisting of primitive objects.

Fixed-Point Spaces

For some recursive domain equation $D \cong F(D)$, the solution is a *limit space*.

- If D is a retract of $F(D)$, then we can characterize D as the space of fixed points of the map defined by the embedding/projection pairs.
- The Knaster-Tarski fixed-point theorem proves that the space of fixed points of a function f on a DCPO is a DCPO. However, this isn't enough for us.
- We have a sequence of spaces $D_0 \subset D_1 \subset D_2 \subset \ldots$, where $D_n = F^n(D_0)$. We want the limit D_ω.

Scott's Inverse Limit Theorem

Scott's *inverse limit theorem* generalizes Knaster-Tarski and completes the construction.

- We know that each D_i is a retract of D_{i+1}, and therefore D_{i+k}.
- For all $i < j$, we therefore have $e_{i,j} = e_j \circ e_{j-1} \circ \ldots e_i$ and $p_{i,j} = p_i \circ p_{i+1} \circ \ldots \circ p_j$
- We define $e_{i,\omega} = \lim_{j \to \omega} e_{i,j}$, but this only maps elements of D_0 to non-$\perp$ values.
- We can define $p_{i,\omega} = \lim_{j \to \omega} p_{i,j}$, which gives us $D_i = p_{i,\omega}(D_\omega)$, where D_ω is the *inverse* limit.

Scott's inverse limit theorem shows that the inverse limit D_ω exists, is unique, and is also the direct limit.

Categorical Constructions

Over time, domain theory has absorbed more and more category theory. Generalized constructions based on categories have replaced the original topological tools.
The key difficulty in applying category theory is contravariant functors. Two approaches deal with this.
Smyth and Plotkin use retracts as the arrows, which gets back to covariance.
Freyd splits the covariant and contravariant parts and uses a construction similar to the implementation of full logic using monotone operators.

Metric Space Semantics

Metric space semantics is a more modern approach, based on metric spaces. It is similar in structure, and has some advantages over Scott-domains:

- No inverse limit theorem, uses Banach's Fixed Point Theorem.
- Closer conceptually to analysis.
- Tends to work better for formulating memories, separation logic.

However, it also has some disadvantages:

- Can't handle non-metrizable spaces.
- Can be harder to visualize and explain.

Metric Spaces

A *metric space* is a set equipped with a distance function d with the following properties:

- $d(x, x) = 0$, $d(x, y) > 0$ if $x \neq y$
- $d(x, y) = d(y, x)$
- $d(x, z) \leq d(x, y) + d(y, z)$

The following are basic definitions in metric spaces:

- An ϵ-*ball* around a point x is the set of all points of distance at most ϵ from x.
- A function f is *continuous* if for any ϵ-ball Y around $f(x)$, there exists a δ-ball X around x such that $Y = f(X)$.
- A *limit* of a sequence $x_1, x_2, \ldots$ is a point x such that for any ϵ, there exists some k such that for all $i > k$, $d(x, x_i) < \epsilon$.

Metric Space Semantics

In metric space semantics, we represent incomplete terms as an ϵ-ball containing all the possible values the term might take.

- Refining the term shrinks the ball more and more, until it shrinks down to a single point.
- The analog of $\perp$ is the set containing the entire space.

We also have a theorem strong enough to build limit spaces:

- A given map F from A to B is *contractive* if $d(F(x), F(y)) < d(x, y)$
- Banach's Fixed Point Theorem says any *contractive* map has a unique fixed point.
- Thus, we need to show that our constructions give rise to contractive maps.

Ultrametric Spaces and Contraction Maps

*Ultra*metric spaces[1] are a special kind of metric space in which all the usual constructions behave nicely.

- In an ultrametric space, we replace the triangle inequality with $d(x,z) \leq \max[d(x,y), d(y,z)]$
- Cartesian products, continuous functions, et. al. are *nonexpansive* maps ($d(F(x), F(y)) \leq d(x,y)$).
- There is a well-behaved map $\frac{1}{2}$, which halfs all the distances (more generally, multiplies all distances by any $k > 0$).
- The map $\frac{1}{2}$ turns any non-expansive map into a contractive map.

[1] Actually, complete, 1-bounded, non-empty ultrametric spaces.

Takeaways

- Scott's characterization of semi-definite terms as a partial order is a very powerful abstraction.
- Used in many other areas: abstract interpretation, propagators, others.
- Non-monotone, discontinuous phenomena can be characterized using monotone continuous functions.
- This forms an elegant, effective foundation for approaches to many hard problems.
- There are deep connections between computational theory and higher mathematics.

Final takeaway: denotational semantics is intimidating at first, but actually not that bad if you know where it's leading.