Denotational Semantics

Based on a lecture by

Martin Abadi

Introduction

- Denotational semantics is supposed to be mathematical:
  - The meaning of an expression is a mathematical object
  - A fair amount of mathematics is involved
- Denotational semantics is compositional
- Denotational semantics is more abstract and canonical than operational semantics
  - No small step vs. big step
- Denotational semantics is also called
  - Fixed point semantics
  - Mathematical semantics
  - Scott-Strachey semantics

Plan

- Definition of the denotational semantics of While (first attempt)
- Complete partial orders and related properties
  - Montonicity
  - Continuity
- Definition of denotational semantics of While

Denotational semantics

- **A:** Aexp $\rightarrow$ ($\Sigma\rightarrow$N)
- **B**: Bexp $\rightarrow$($\Sigma\rightarrow$T)
- **S:** Stm $\rightarrow$($\Sigma\rightharpoonup\Sigma$)
- Defined by structural induction

Denotational semantics of Aexp

- **A:** Aexp $\to$ ($\Sigma \to$N)
- **A** $[\![n]\!] = \{(\sigma, n) \mid \sigma \in \Sigma\}$
- **A** $[\![X]\!] = \{(\sigma, \sigma\, X) \mid \sigma \in \Sigma\}$
- **A** $[\![a_0+a_1]\!] = \{(\sigma, n_0+n_1) \mid (\sigma, n_0) \in \mathbf{A}[\![a_0]\!], (\sigma, n_1) \in \mathbf{A}[\![a_1]\!]\}$
- **A** $[\![a_0-a_1]\!] = \{(\sigma, n_0-n_1) \mid (\sigma, n_0) \in \mathbf{A}[\![a_0]\!], (\sigma, n_1) \in \mathbf{A}[\![a_1]\!]\}$
- **A** $[\![a_0 \times a_1]\!] = \{(\sigma, n_0 \times n_1) \mid (\sigma, n_0) \in \mathbf{A}[\![a_0]\!], (\sigma, n_1) \in \mathbf{A}[\![a_1]\!]\}$

Lemma: A $[\![a]\!]$ is a function

Denotational semantics of Aexp with λ

- **A:** Aexp $\rightarrow$ ($\Sigma \rightarrow$N)
- **A** $[\![n]\!] = \lambda\sigma\in\Sigma.n$
- **A** $[\![X]\!] = \lambda\sigma\in\Sigma.\sigma(X)$
- **A** $[\![a_0+a_1]\!] = \lambda\sigma\in\Sigma.(\mathbf{A}\,[\![a_0]\!]\sigma+\mathbf{A}[\![a_1]\!]\sigma)$
- **A** $[\![a_0-a_1]\!] = \lambda\sigma\in\Sigma.(\mathbf{A}\,[\![a_0]\!]\sigma-\mathbf{A}[\![a_1]\!]\sigma)$
- **A** $[\![a_0\times a_1]\!] = \lambda\sigma\in\Sigma.(\mathbf{A}\,[\![a_0]\!]\sigma\times\mathbf{A}[\![a_1]\!]\sigma)$

Denotational semantics of Bexp

- $\mathbf{B}$: Bexp $\to (\Sigma \to T)$
- $\mathbf{B}[\![\text{true}]\!] = \{(\sigma, \text{true}) \mid \sigma \in \Sigma\}$
- $\mathbf{B}[\![\text{false}]\!] = \{(\sigma, \text{false}) \mid \sigma \in \Sigma\}$
- $\mathbf{B}[\![a_0 = a_1]\!] = \{(\sigma, \text{true}) \mid \sigma \in \Sigma \;\&\; \mathbf{A}[\![a_0]\!]\sigma = \mathbf{A}[\![a_1]\!]\sigma\} \cup$
  $\{(\sigma, \text{false}) \mid \sigma \in \Sigma \;\&\; \mathbf{A}[\![a_0]\!]\sigma \neq \mathbf{A}[\![a_1]\!]\sigma\}$
- $\mathbf{B}[\![a_0 \leq a_1]\!] = \{(\sigma, \text{true}) \mid \sigma \in \Sigma \;\&\; \mathbf{A}[\![a_0]\!]\sigma \leq \mathbf{A}[\![a_1]\!]\sigma\} \cup$
  $\{(\sigma, \text{false}) \mid \sigma \in \Sigma \;\&\; \mathbf{A}[\![a_0]\!]\sigma \not\leq \mathbf{A}[\![a_1]\!]\sigma\}$
- $\mathbf{B}[\![\neg b]\!] = \{(\sigma, \neg_T t) \mid \sigma \in \Sigma, (\sigma, t) \in \mathbf{B}[\![b]\!]\}$
- $\mathbf{B}[\![b_0 \wedge b_1]\!] = \{(\sigma, t_0 \wedge_T t_1) \mid \sigma \in \Sigma, (\sigma, t_0) \in \mathbf{B}[\![b_0]\!], (\sigma, t_1) \in \mathbf{B}[\![b_1]\!]\}$
- $\mathbf{B}[\![b_0 \vee b_1]\!] = \{(\sigma, t_0 \vee_T t_1) \mid \sigma \in \Sigma, (\sigma, t_0) \in \mathbf{B}[\![b_0]\!], (\sigma, t_1) \in \mathbf{B}[\![b_1]\!]\}$

Lemma: $B[\![b]\!]$ is a function

Denotational semantics of statements?

- Running a statement s starting from a state σ yields another state σ’
- So, we may try to define **S** ⟦s⟧ as a function that maps σ to σ’:
  - **S** ⟦.⟧: Stm →(Σ → Σ)

Denotational semantics of commands?

- Problem: running a statement might not yield anything if the statement does not terminate
- We introduce the special element $\perp$ to denote a special outcome that stands for non-termination
- For any set X, we write $X_\perp$ for $X \cup \{\perp\}$

- Convention:
  - whenever $f \in X \to X_\perp$ we extend f to $X_\perp \to X_\perp$ "strictly" so that $f(\perp) = \perp$

Denotational semantics of statements?

- We try:
  - $S [\![.]\!] : Stm \to (\Sigma_\perp \to \Sigma_\perp)$
- $S [\![skip]\!] \sigma = \sigma$
- $S [\![s_0 ; s_1]\!] \sigma = S [\![s_1]\!] (S [\![s_0]\!] \sigma)$
- $S [\![\text{if } b \text{ then } s_0 \text{ else } s_1]\!] \sigma =$

  $\text{if } B [\![b]\!] \sigma \text{ then } S [\![s_0]\!] \sigma \text{ else } S [\![s_1]\!] \sigma$

Examples

- $S [\![X:= 2;\ X:=1]\!] \sigma = \sigma[X \mapsto 1]$
- $S [\![\text{if true then } X:=2;\ X:=1 \text{ else } \ldots]\!] \sigma = \sigma[X \mapsto 1]$
- The semantics does not care about intermediate states
- So far, we did not explicitly need $\perp$

Denotational semantics of loops?

- $S [\![\text{while b do s}]\!] \sigma = ?$

Denotational semantics of statements?

- Abbreviation $W = S[\![\text{while b do s}]\!]$
- Idea: we rely on the equivalence

  while b do s ~ if b then (s; while b do s) else skip
- We may try using unwinding equation

  $W(\sigma) = \text{if } B[\![b]\!]\sigma \text{ then } W(S[\![s]\!]\,\sigma) \text{ else } \sigma$
- Unacceptable solution
  - Defines W in terms of itself
  - It not evident that a suitable W exists
  - It may not describe W uniquely (e.g., for while true do skip)

Introduction to Domain Theory

- We will solve the unwinding equation through a general theory of recursive equations
- Think of programs as processors of streams of bits (streams of 0’s and 1’s, possibly terminated by $) What properties can we expect?

Motivation

- Let “isone” be a function that must return “1\$” when the input string has at least a 1 and “0\$” otherwise
  - isone(00…0\$) = 0\$
  - isone(xx…1…\$) =1\$
  - isone(0…0) =?
- **Monotonicity** : Output is never retracted
  - More information about the input is reflected in more information about the output
- How do we express monotonicity precisely?

Montonicity

- Define a partial order
  $x \sqsubseteq y$
  - A partial order is reflexive, transitive, and antisymmetric
  - y is a refinement of x
- For streams of bits $x \sqsubseteq y$ when x is a prefix of y
- For programs, a typical order is:
  - No output (yet) $\sqsubseteq$ some output

Montonicity

- A set equipped with a partial order is a poset
- Definition:
  - D and E are postes
  - A function f: D $\rightarrow$E is monotonic if $\forall x, y \in D: x \sqsubseteq_D y \Rightarrow f(x) \sqsubseteq_E f(y)$
  - The semantics of the program ought to be a monotonic function
    - More information about the input leads to more information about the output

Montonicity Example

- Consider our "isone" function with the prefix ordering
- Notation:
  - 0^k is the stream with k consecutive 0's
  - 0^∞ is the infinite stream with only 0's
- Question (revisited): what is isone(0^k)?
  - By definition, isone(0^k\$) = 0\$ and isone(0^k1\$) = 1\$
  - But $0^k \sqsubseteq 0^k\$$ and $0^k \sqsubseteq 0^k1\$$
  - "isone" must be monotone, so:
    - isone(0^k) $\sqsubseteq$ isone(0^k\$) = 0\$
    - isone(0^k) $\sqsubseteq$ isone(0^k1\$) = 1\$
  - Therefore, monotonicity requires that isone(0^k) is a common prefix of 0\$ and 1\$, namely ε

Motivation

- Are there other constraints on “isone”?
- Define “isone” to satisfy the equations
  - isone(ε)=ε
  - isone(1s)=1$
  - isone(0s)=isone(s)
  - isone($)=0$
- What about 0^∞?
- Continuity: finite output depends only on finite input (no infinite lookahead)

Chains

- A chain is a countable increasing sequence $\langle x_i \rangle = \{x_i \in X \mid x_0 \sqsubseteq x_1 \sqsubseteq \ldots \}$
- An upper bound of a set if an element "bigger" than all elements in the set
- The least upper bound is the "smallest" among upper bounds:
  – $x_i \sqsubseteq \sqcup \langle x_i \rangle$ for all $i \in N$
  – $\sqcup \langle x_i \rangle \sqsubseteq y$ for all upper bounds y of $\langle x_i \rangle$ and it is unique if it exists

Complete Partial Orders

0 1 2 …

⊥

- Not every poset has an upper bound
  - with $\bot \sqsubseteq n$ and $n \sqsubseteq n$ for all $n \in N$
  - {1, 2} does not have an upper bound
- Sometimes chains have no upper bound

⋮

2

1

0

The chain

$0 \leq 1 \leq 2 \leq \ldots$

does not have an upper bound

Complete Partial Orders

- It is convenient to work with posets where every chain (not necessarily every set) has a least upper bound
- A partial order P is complete if every chain in P has a least upper bound also in P
- We say that P is a complete partial order (cpo)
- A cpo with a least ("bottom") element ⊥ is a pointed cpo (pcpo)

Examples of cpo's

- Any set P with the order
  $x \sqsubseteq y$ if and only if $x = y$ is a cpo
  It is discrete or flat
- If we add $\perp$ so that $\perp \sqsubseteq x$ for all $x \in P$, we get a flat pointed cpo
- The set N with $\leq$ is a poset with a bottom, but not a complete one
- The set $N \cup \{ \infty \}$ with $n \leq \infty$ is a pointed cpo
- The set N with $\geq$ is a cpo without bottom
- Let S be a set and P(S) denotes the set of all subsets of S ordered by set inclusion
  - P(S) is a pointed cpo

Constructing cpos

- If D and E are pointed cpos, then so is $D \times E$

  $(x, y) \sqsubseteq_{D \times E} (x', y')$ iff $x \sqsubseteq_D x'$ and $y \sqsubseteq_E y'$

  $\bot_{D \times E} = (\bot_D , \bot_E)$

  $\bigsqcup (x_i , y_i) = (\bigsqcup_D x_i , \bigsqcup_E y_i)$

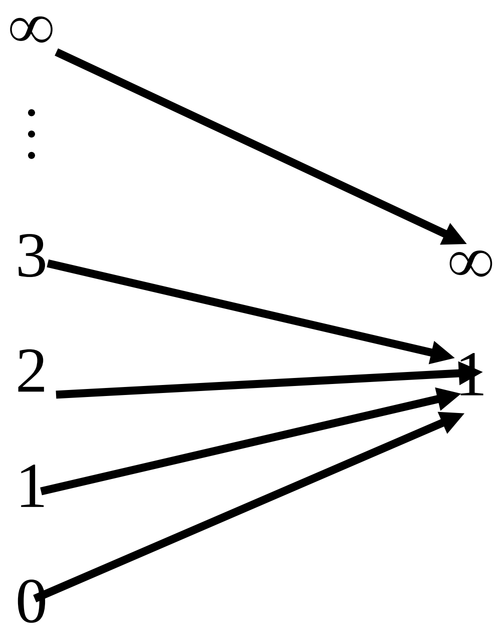
Constructing cpos (2)

- If S is a set of E is a pcpos, then so is
  $S \rightarrow E$
  $m \sqsubseteq m'$ iff $\forall s \in S$: $m(s) \sqsubseteq_E m'(s)$
  $\bot_{S \rightarrow E} = \lambda s.\ \bot_E$
  $\sqcup (m , m') = \lambda s.m(s) \sqcup_E m'(s)$

Continuity

- A monotonic function maps a chain of inputs into a chain of outputs:
  $x_0 \sqsubseteq x_1 \sqsubseteq \ldots \Rightarrow f(x_0) \sqsubseteq f(x_1) \sqsubseteq \ldots$
- It is always true that:
  $\sqcup_i \langle f(x_i) \rangle \sqsubseteq f(\sqcup_i \langle x_i \rangle)$
- But
  $f(\sqcup_i \langle x_i \rangle) \sqsubseteq \sqcup_i \langle f(x_i) \rangle$
  is not always true

A Discontinuity Example



$f(\sqcup_i <x_i>) \neq \sqcup i <f(x_i)>$

Continuity

- Each $f(x_i)$ uses a “finite” view of the input
- $f(\sqcup < x_i >)$ uses an “infinite” view of the input
- A function is continuous when
  $f(\sqcup < xi >) = \sqcup_i < f(x_i) >$
- The output generated using an infinite view of the input does not contain more information than all of the outputs based on finite inputs
- Scott’s thesis: The semantics of programs can be described by a continuous functions

Examples of Continuous Functions

- For the partial order ($N \cup \{\infty\}, \leq$)
  - The identity function is continuous
    $id(\sqcup n_i) = \sqcup id(n_i)$
  - The constant function "five(n)=5" is continuous
    $five(\sqcup n_i) = \sqcup five(n_i)$
  - If $isone(0^{\infty}) = \varepsilon$ then isone is continuos
- For a flat cpo A, any monotonic function
  $f: A_{\perp} \rightarrow A_{\perp}$
  such that f is strict is continuous
- Chapter 8 of the Wynskel textbook includes many more continuous functions

Fixed Points

- Solve the equation:

$$W(\sigma) = \begin{cases} W(S[\![s]\!]\,\sigma) & \text{if } B[\![b]\!](\sigma)=\text{true} \\ \sigma & \text{if } B[\![b]\!](\sigma)=\text{false} \\ \bot & \text{if } B[\![b]\!](\sigma)=\bot \end{cases}$$

  where $W: \Sigma_\bot \to \Sigma_\bot$
  $W = S[\![\text{while be do s}]\!]$
- This equation can be written as W = F(W) with:
- $$F(W) = \lambda\sigma. \begin{cases} W(S[\![s]\!]\,\sigma) & \text{if } B[\![b]\!](\sigma)=\text{true} \\ \sigma & \text{if } B[\![b]\!](\sigma)=\text{false} \\ \bot & \text{if } B[\![b]\!](\sigma)=\bot \end{cases}$$

Fixed Point (cont)

- Thus we are looking for a solution for W = F(W)
  - a fixed point of F
- Typically there are many fixed points
- We may argue that W ought to be continuous $W \in [\Sigma_{\perp} \rightarrow \Sigma_{\perp}]$
- Cut the number of solutions
- We will see how to find the least fixed point for such an equation provided that F itself is continuous

Fixed Point Theorem

- Define $F^k = \lambda x.\, F(F(\ldots F(x)\ldots))$ (F composed k times)
- If D is a pointed cpo and $F : D \rightarrow D$ is continuous, then
  - for any fixed-point x of F and $k \in N$
    $F^k(\bot) \sqsubseteq x$
  - The least of all fixed points is
    $\sqcup_k F^k(\bot)$
- Proof:

  i. By induction on k.
     - Base: $F^0(\bot) = \bot \sqsubseteq x$
     - Induction step: $F^{k+1}(\bot) = F(F^k(\bot)) \sqsubseteq F(x) = x$

  ii. It suffices to show that $\sqcup_k F^k(\bot)$ is a fixed-point
     - $F(\sqcup_k F^k(\bot)) = \sqcup_k F^{k+1}(\bot) = \sqcup_k F^k(\bot)$

Fixed-Points (notes)

- If F is continuous on a pointed cpo, we know how to find the least fixed point
- All other fixed points can be regarded as refinements of the least one
- They contain more information, they are more precise
- In general, they are also more arbitrary
- They also make less sense for our purposes

Denotational Semantics of While

- $\Sigma_{\perp}$ is a flat pointed cpo
  - A state has more information on non-termination
  - Otherwise, the states must be equal to be comparable (information-wise)
- We want strict functions $\Sigma_{\perp} \to \Sigma_{\perp}$ (therefore, continuous functions)
- The partial order on $\Sigma_{\perp} \to \Sigma_{\perp}$
  $f \sqsubseteq g$ iff $f(x) = \perp$ or $f(x) = g(x)$ for all $x \in \Sigma_{\perp}$
  - g terminates with the same state whenever f terminates
  - g might terminate for more inputs

Denotational Semantics of While

- Recall that W is a fixed point of
  $F:[[\Sigma_\bot \rightarrow \Sigma_\bot] \rightarrow [\Sigma_\bot \rightarrow \Sigma_\bot]]$

$$F(w) = \lambda\sigma. \begin{cases} w(S[\![s]\!](\sigma)) & \text{if } B[\![b]\!](\sigma)=\text{true} \\ \sigma & \text{if } B[\![b]\!](\sigma)=\text{false} \\ \bot & \text{if } B[\![b]\!](\sigma)=\bot \end{cases}$$

- F is continuous
- Thus, we set
  $S[\![\text{while b do c}]\!] = \sqcup F^k(\bot)$
- Least fixed point
  - Terminates least often of all fixed points
- Agrees on terminating states with all fixed point

Denotational Semantics of While

- $S [\![skip]\!] = \lambda\sigma.\sigma$
- $S [\![X := exp]\!] = \lambda\sigma.\sigma[X \mapsto A[\![exp]\!] \sigma]$
- $S [\![s_0 ; s_1]\!] = \lambda\sigma. S [\![s_1]\!] (S [\![s_0]\!] \sigma)$
- $S [\![\text{if } b \text{ then } s_0 \text{ else } s_1]\!] =$

  $\lambda\sigma. \text{ if } B[\![b]\!] \sigma \text{ then } S [\![s_0]\!] \sigma \text{ else } S [\![s_1]\!] \sigma$
- $S [\![\text{ while } b \text{ do } s]\!] = \sqcup F^k(\bot)$ k=0, 1, … where

  $F = \lambda w. \lambda\sigma. \text{ if } B[\![b]\!](\sigma)=\text{true } w(S[\![s]\!](\sigma)) \text{ else } \sigma$

Example(1)

- while true do skip
- $F:[[\Sigma_\perp \rightarrow \Sigma_\perp] \rightarrow [\Sigma_\perp \rightarrow \Sigma_\perp]]$

$$F = \lambda w.\lambda\sigma.\begin{cases} w(S[\![s]\!](\sigma)) & \text{if } B[\![b]\!](\sigma)=\text{true} \\ \sigma & \text{if } B[\![b]\!](\sigma)=\text{false} \\ \perp & \text{if } B[\![b]\!](\sigma)=\perp \end{cases}$$

$B[\![\text{true}]\!]=\lambda\sigma.\text{true}$

$S[\![\text{skip}]\!]=\lambda\sigma.\sigma$

$$F = \lambda\, w.\lambda\sigma.w(\sigma)$$

$$F^0(\perp)=\perp \quad \sqcup \quad F^1(\perp)=\perp \quad \sqcup \quad F^2(\perp)=\perp \qquad\qquad \perp$$

Example(2)

- while false do s
- $F:[[\Sigma_\perp \to \Sigma_\perp] \to [\Sigma_\perp \to \Sigma_\perp]]$

$$F = \lambda w.\lambda\sigma. \begin{cases} w(S[\![s]\!](\sigma)) & \text{if } B[\![b]\!](\sigma)=\text{true} \\ \sigma & \text{if } B[\![b]\!](\sigma)=\text{false} \\ \perp & \text{if } B[\![b]\!](\sigma)=\perp \end{cases}$$

$B[\![\text{false}]\!]=\lambda\sigma.\text{false}$

$$F = \lambda w.\lambda\sigma.\sigma$$

$F^0(\perp)= \perp \quad \sqcup \quad F^1(\perp) = \lambda\sigma.\sigma \sqcup \quad F^2(\perp) = \lambda\sigma.\sigma \qquad \lambda\sigma.\sigma$

Example(3)

$[\![\text{while } x \neq 3 \text{ do } x = x - 1]\!] = \sqcup F^k(\bot) \quad k=0, 1, \ldots$

where

$F = \lambda w. \lambda\sigma. \text{ if } \sigma(x) \neq 3 \ w(\sigma[x \mapsto \sigma(x) - 1]) \text{ else } \sigma$

$F^{0}(\bot)$	$\bot$
$F^1(\bot)$	if $\sigma(x) \neq 3 \ \bot(\sigma[x \mapsto \sigma(x) - 1])$ else σ
	if $\sigma(x) \neq 3$ then $\bot$ else σ
$F^2(\bot)$	if $\sigma(x) \neq 3$ then $F^1(\sigma[x \mapsto \sigma(x) - 1])$ else σ
	if $\sigma(x) \neq 3$ then (if $\sigma[x \mapsto \sigma(x) - 1]\ x \neq 3$ then $\bot$ else $\sigma[x \mapsto \sigma(x) - 1]$) else σ
	if $\sigma(x) \neq 3$ (if $\sigma(x) \neq 4$ then $\bot$ else $\sigma[x \mapsto \sigma(x) - 1]$) else σ
	if $\sigma(x) \in \{3, 4\}$ then $\sigma[x \mapsto 3]$ else $\bot$
$F^k(\bot)$	if $\sigma(x) \in \{3, 4, \ldots k\}$ then $\sigma[x \mapsto 3]$ else $\bot$
lfp(F)	if $\sigma(x) \geq 3$ then $\sigma[x \mapsto 3]$ else $\bot$

Example 4 Nested Loops

```
S ==
Z := 0 ;
while X > 0 do (
    Y := X;
    while (Y>0) do
        Z := Z + Y ;
        Y: =  Y- 1; )
  X = X – 1
  )
```

s⟦inner-loop⟧=

$$\begin{cases} [Y \mapsto 0][Z \mapsto \sigma(Z)+\sigma(Y) * (\sigma(Y)+1)/2] & \text{if } \sigma(Y)\geq 0 \\ \bot & \text{if } \sigma(Y)<0 \end{cases}$$

s⟦outer-loop⟧=

$$\begin{cases} [Y \mapsto 0][X \mapsto 0][Z \mapsto \sigma(Z)+\sigma(X) \times (\sigma(X) + 1) \times (1 + (2\sigma(X) + 1)/3)/4\ ] & \text{if } \sigma(X)\geq 0 \\ \bot & \text{if } \sigma(X)<0 \end{cases}$$

s⟦S⟧=

$$\begin{cases} [Y \mapsto 0][X \mapsto 0][Z \mapsto \sigma(X) \times (\sigma(X) + 1) \times (1 + (2\sigma(X) + 1)/3)/4\ ] & \text{if } \sigma(X)\geq 0 \\ \bot & \text{if } \sigma(X)<0 \end{cases}$$

Equivalence of Semantics

- $\forall \sigma, \sigma' \in \Sigma$:
  $\sigma' = S[\![s]\!]\sigma \Leftrightarrow \langle s, \sigma \rangle \rightarrow \sigma' \Leftrightarrow \langle s, \sigma \rangle \Rightarrow^* \sigma'$

Complete Partial Orders

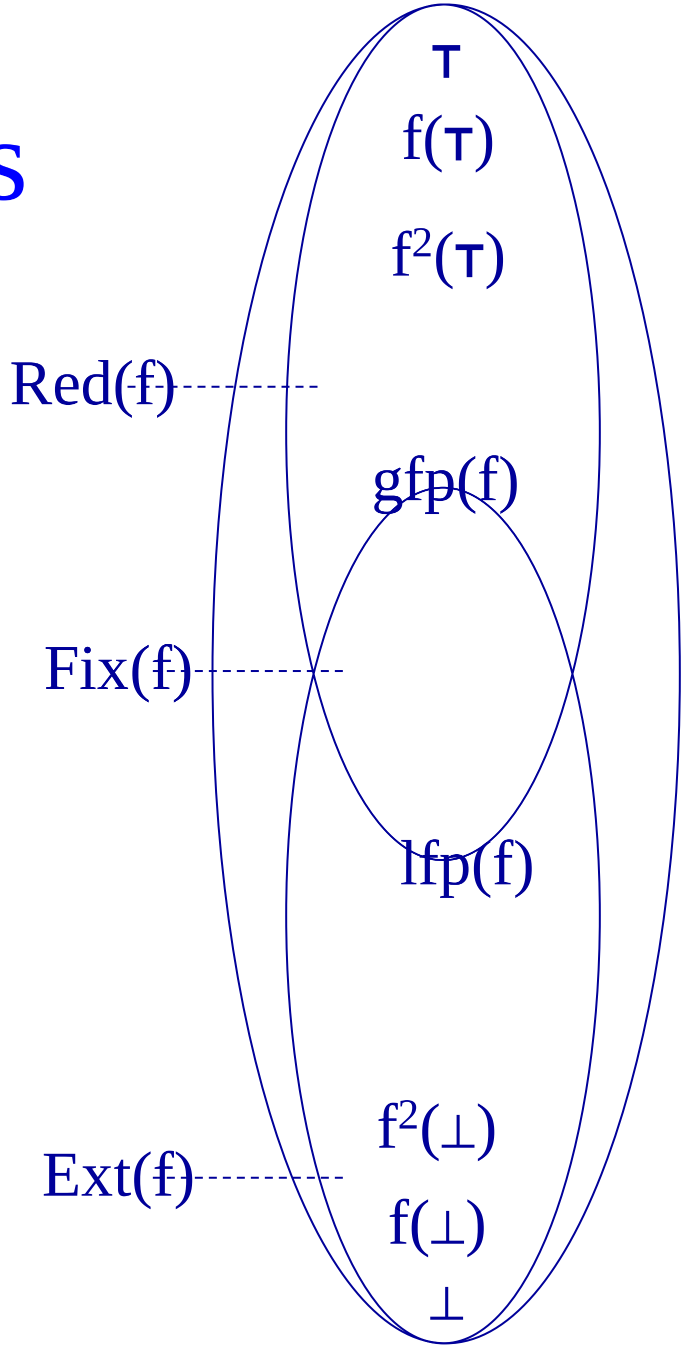
- Let (D, $\sqsubseteq$) be a partial order
  - D is a complete lattice if every subset has both greatest lower bounds and least upper bounds

Knaster-Tarski Theorem

- Let f: L $\rightarrow$L be a monotonic function on a complete lattice L
- The least fixed point lfp(f) exists
  – lfp(f) = $\sqcap\{x \in L: f(x) \sqsubseteq x\}$

Fixed Points

- A monotone function f: L $\rightarrow$ L where (L, $\sqsubseteq$, $\sqcup$, $\sqcap$, $\bot$, $\top$) is a complete lattice
- Fix(f) = { l: l $\in$ L, f(l) = l}
- Red(f) = {l: l $\in$ L, f(l) $\sqsubseteq$ l}
- Ext(f) = {l: l $\in$ L, l $\sqsubseteq$ f(l)}
  - $l_1 \sqsubseteq l_2 \Rightarrow f(l_1) \sqsubseteq f(l_2)$
- Tarski's Theorem 1955: if f is monotone then:
  - lfp(f) = $\sqcap$ Fix(f) = $\sqcap$ Red(f) $\in$ Fix(f)
  - gfp(f) = $\sqcup$ Fix(f) = $\sqcup$ Ext(f) $\in$ Fix(f)



Summary

- Denotational definitions are not necessarily better than operational semantics, and they usually require more mathematical work
- The mathematics may be done once and for all
- The mathematics may pay off:
- Some of its techniques are being transferred to operational semantics.
- It is trivial to prove that
  “If $B[\![b_1]\!] = B[\![b_2]\!]$ and $C[\![c_1]\!] = C[\![c_2]\!]$
  then
  $C[\![\text{while } b_1 \text{ do } c_1]\!] = C[\![\text{while } b_2 \text{ do } c_2]\!]$”
  (compare with the operational semantics)

Summary

- Denotational semantics provides a way to declare the meaning of programs in an abstract way
  - Can handle
    - side-effects
    - loops
    - Recursion
    - Gotos
    - non-determinism
  - But not low level concurrency
- Fixed point theory provides a declarative way to specify computations
  - Many usages