Introduction to Axiomatic Semantics

Meeting 9, CSCI 5535, Spring 2009

Announcements

- Homework 3 is out, due Mon Feb 16
  - No domain theory!
- Homework 1 is graded
  - Feedback attached
  - 14.2 (mean), 13 (median), out of 18 total
  - Congratulations Moss for breaking 5 interpreters!
- Supplementary reading on denotational semantics and domain theory

Survey from Homework 2

- Time spent: 4.3hrs (mean), 4hrs (median)
  - less than Homework 1!
- Discussion great but could be more directed
  - Raise hands when I ask discussion question
  - Still ok to shout out: "Wait, I don't understand that!", "Slow down!"

Questions?

Review of Denotational Semantics and Domain Theory

DS Glossary

- A partial order: x ≤ y (for x, y ∈ D)
  - Reflexive, transitive, and anti-symmetric
- A poset (partially ordered set): (D, ≤)
- A lub (least upper bound): lubX
  - For all x ∈ X, x ≤ lubX (upper bound)
  - For any y s.t. (∀x ∈ X, x ≤ y), we have lubX ≤ y (least)
- A chain (ω-chain): x_1 ≤ ... ≤ x_n ≤ ...
- A cpo (complete partial order):
  - A poset where all chains have least upper bounds
- A cpo with bottom: Most important structure!
  - A cpo with a least element ⊥ called “bottom”
DS Glossary: Related Structures

• A lattice:
  - A poset where each pair \( x, y \) has a lub \( x \lor y \) (a join) and glb (greatest lower bound) \( x \land y \) (a meet)
  - Equivalently, all finite subsets have lubs and glbs
  - Variants: join-semi-lattice, meet-semi-lattice

• A complete lattice:
  - A poset where all subsets have lubs and glbs

• A domain:
  - Generic term for these structures.
  - For us, domain = cpo with bottom

DS Glossary

• Let \((D, \sqsubseteq)\) be a domain:
• A function \( F : D_1 \to D_2 \) is monotonic iff

\[ \forall x, y \in D_1. x \sqsubseteq y \Rightarrow F(x) \sqsubseteq F(y) \]

• A function \( F : D_1 \to D_2 \) is continuous iff

\[ \forall \text{ chains } x_i \text{ in } D_1: F(\bigsqcup_i x_i) = \bigsqcup_i (F(x_i)) \]

Recall: Information Ordering on Funcs

• Let \( f, g \in \mathbb{Z} \to \mathbb{Z}_\bot \)
• Define \( f \sqsubseteq g \) as

\[ \forall x \in \mathbb{Z}. f(x) = \bot \text{ or } f(x) = g(x) \]

- We say that \( g \) refines \( f \)
- We say that \( f \) approximates \( g \)
- We say that \( g \) provides more information than \( f \)

Recall: Semantic Functional for while

• The meaning of a context is a semantic functional
  \[ F : (\mathbb{Z} \to \mathbb{Z}_\bot) \to (\mathbb{Z} \to \mathbb{Z}_\bot) \]
  such that

\[ F[K[w]] = F[w] \]

• For "while": \( K = \text{if } b \text{ then } c; \text{ else skip} \)

\[ F[w x] = \text{if } [b] x \text{ then } w ([c] x) \text{ else } x \]

• Unwinding equation can be rewritten as

\[ W = F W \]
A Property of $F$

- Consider $f_0 \sqsubseteq f_1$. What can we say about the relationship between $F f_0 x$ and $F f_1 x$, for any $x$?

Monotonicity

- Consider $f_0 \sqsubseteq f_1$. What can we say about the relationship between $F f_0 x$ and $F f_1 x$, for any $x$?

- Assume $F f_0 x = n \neq \bot$. Show that $F f_1 x = n$.
  - In computing $F f_0 x$, $f_0$ is invoked a finite number of times.
  - All those invocations terminate with some values.
  - The value of $f_0$ at other points does not matter!
  - But $f_1$ terminates with some results everywhere $f_0$ terminates.
  - Thus $F f_1 x = n$ (determinism of $F$).

- If $F f_0 x = \bot$, it could be that $F f_1 x \neq \bot$.
  - Take $F f x = f x$, $f_0 (0) = \bot$, and $f_1 (0) = 0$.

- In general, if $f_0 \sqsubseteq f_1$, then $F f_0 \sqsubseteq F f_1$.

F must be monotonic.

The Fixed-Point Theorem

- If $F$ is a semantic function corresponding to a context in our language, $F$ is monotonic and continuous (we assert).
  - For any fixed-point $G$ of $F$ and $k \in \mathbb{N}$.
  - $F^k (\lambda x. \bot) \sqsubseteq G$.
    - The least of all fixed points is $\biguparrow_k F^k (\lambda x. \bot)$.
  - Proof (not detailed in the lecture):
    1. By mathematical induction on $k$.
      - Base: $F^0 (\lambda x. \bot) = \lambda x. \bot \sqsubseteq G$.
      - Inductive: $F^{k+1} (\lambda x. \bot) = F (F^k (\lambda x. \bot)) \sqsubseteq F (G) \sqsubseteq G$.
    - Sufﬁces to show that $\biguparrow_k F^k(x. \bot)$ is a fixed-point.
      - $F \biguparrow_k F^k (\lambda x. \bot) = \biguparrow_k F^2 (\lambda x. \bot) = \biguparrow_k F^k (\lambda x. \bot)$.
        - by continuity

Consequences of the Fixed-Point Theorem

1) Infinite pass $\rightarrow$ while $\Rightarrow$ ordinary or least $\Rightarrow$ unique denot for while
2) Computational flavor $\Rightarrow$ how to compute $\text{ifp}$ (least fixed points)

Recall: Learning Goals

- Key features of denotational semantics
  - Compositional
  - Meaning is a "math object"
  - When to use DS?

- DS uses $\bot$ ("bottom") to mean non-termination
- DS uses fixed points and domains to handle while
  - This is the tricky bit

Remember SLAM and BLAST?

End of Review

On to Homework 3 Hints
How’s the Homework Going?

• Remember that you cannot just define a meaning function in terms of itself
• You must use some fixed point machinery

Meta-Comment

• A key part of doing research is noticing when something is incongruous or when something changes - or otherwise spotting patterns.

Meta-Comment

• A key part of doing research is noticing when something is incongruous or when something changes - or otherwise spotting patterns.

What’s Wrong Here?

\[
\text{char} :: \text{str} \cup \text{characters} \\
\text{s} = r_1 :: r_2 :: s \\
\not\text{r}_1 \text{r}_2 \text{ matches } s \text{ leaving } S
\]

Plan for Axiomatic Semantics

• History and Motivation
• Assertions
• Validity
• Derivation Rules
• Soundness
• Completeness

End of Homework Hints

On to Axiomatic Semantics
Review by Class Participation

- Tell Me About Operational Semantics
  
  execution = interpreter + command

- Tell Me About Structural Induction
  on command
  on derivations

- Tell Me About Denotational Semantics
  "real", compositional

Axiomatic Semantics

A semantics that is appropriate for arguing program correctness

One-Slide Summary

- An axiomatic semantics consists of:
  - A language for stating assertions about programs,
  - Rules for establishing the truth of assertions

- Some typical kinds of assertions:
  - This program terminates
  - If this program terminates, the variables x and y have the same value throughout the execution of the program
  - The array accesses are within the array bounds

- Some typical languages of assertions
  - First-order logic
  - Other logics (temporal, linear, pointer-assertion)
  - Special-purpose specification languages (SLIC, Z, Larch)

History

- Program verification is almost as old as programming (e.g., Checking a Large Routine, Turing 1949)
- In the late '60s, Floyd had rules for flow-charts and Hoare for structured languages
- Since then, there have been axiomatic semantics for substantial languages, and many applications
  - ESC/Java, SLAM, PCC, SPARK Ada, ...

Tony Hoare Quote

"Thus the practice of proving programs would seem to lead to solution of three of the most pressing problems in software and programming, namely, reliability, documentation, and compatibility. However, program proving, certainly at present, will be difficult even for programmers of high caliber; and may be applicable only to quite simple program designs."


Agree or Disagree?
Edsger Dijkstra Quote

"Program testing can be used to show the presence of bugs, but never to show their absence!"

For Next Time

- Homework 3 due Mon Feb 16