This Lecture

- Introduction to Denotational Semantics
- The relation between Operational and Denotational Semantics

Denotational Semantics (1)

Operational Semantics (review):
- The meaning of a term is the term it (ultimately) reduces to, if any:
  - Stuck terms
  - Infinite reduction sequences
- No inherent meaning (structure) beyond syntax of terms.
- Focus on behaviour; important aspects of semantics (such as non-termination) emerges from the behaviour.

Denotational Semantics (2)

- Idea: Semantic function maps (abstract) syntax directly to meaning in a semantic domain.
- Domains consist of appropriate semantic objects (Booleans, numbers, functions, ...) and have structure; in particular, an information ordering.
- The semantic functions are total; in particular, even a diverging computation is mapped to an element in the semantic domain.

Compositionality (1)

- It is usually required that a denotational semantics is compositional: that the meaning of a program fragment is given in terms of the meaning of its parts.
- Compositionality ensures that
  - the semantics is well-defined
  - important meta-theoretical properties hold

Definition

Formally, a denotational semantics for a language $L$ is given by a pair

$$
\langle D, \llbracket \cdot \rrbracket \rangle
$$

where
- $D$ is the semantic domain
- $\llbracket \cdot \rrbracket : L \rightarrow D$ is the valuation function or semantic function.

In simple cases $D$ may be a set, but usually more structure is required leading to domains as defined in domain theory.

Example: Simple Expr. Language (1)

$$
\begin{align*}
&\rightarrow \\
&| \text{true} & \text{constant true} \\
&| \text{false} & \text{constant false} \\
&| \text{if } e \text{ then } e \text{ else } e & \text{conditional} \\
&| 0 & \text{constant zero} \\
&| \text{succ } e & \text{successor} \\
&| \text{pred } e & \text{predecessor} \\
&| \text{iszero } e & \text{zero test}
\end{align*}
$$
Example: Simple Expr. Language (2)

Develop a denotational semantics \( \langle D, [ ] \rangle \) for \( E \), picking \( \mathbb{N} \) as the semantic domain for simplicity:

\[
D = \mathbb{N} \\
[ ] : e \rightarrow \mathbb{N}
\]

However, as there are both Booleans and natural numbers in the object language, a more refined choice for the semantics at the meta level would have been \( \mathbb{N} \sqcup \mathbb{B} \), the disjoint union of natural numbers and Booleans.

(On whiteboard)

Exercises (1)

1. Find the denotation of
   \[ \text{if (iszero (succ 0)) then true else false} \]

Exercises (2)

2. Consider the following language extension:
   \[
   e \rightarrow \text{expressions:} \\
   \ldots \\
   \text{not } e \quad \text{logical negation} \\
   e \& e \quad \text{logical conjunction} \\
   e + e \quad \text{addition} \\
   e - e \quad \text{subtraction} \\
   e \ast e \quad \text{multiplication}
   \]
   Extend the denotational semantics accordingly.

Operational and Denotational Sem. (1)

Given a language \( L \), suppose we have:
- a big-step operational semantics
  \[ \Downarrow \subseteq L \times V \]
  where \( V \subseteq L \) is the set of values
- a denotational semantics
  \[ \langle D, [ ] \rangle \]
  where \( [ ] : L \rightarrow D \)

How should these be related?

Operational and Denotational Sem. (2)

Closed terms \( t_1, t_2 \in L \) are semantically or denotationally equivalent iff

\[ [t_1] = [t_2] \]

Assume \( D \) is ground (no functions; i.e., our closed terms are programs that output something “printable”). We adopt a function

\[ [ ] : D \rightarrow V \]

that maps a semantic value \( d \in D \) to its term representation \( v \in V \).

Operational and Denotational Sem. (3)

Assuming termination:
  - Correctness of operational semantics w.r.t. denotational semantics:
    \[ t \Downarrow v \Rightarrow [t] = [v] \]
  - Completeness of operational semantics w.r.t. denotational semantics:
    \[ [t] = d \Rightarrow t \Downarrow d \]