Formal Semantics

Yegor Bugayenko

Lecture #4 out of 10
80 minutes

The slidedeck was presented by the author in this YouTube Video

While syntax is a representation of a program, semantics $S(P)$ is a formal description of execution of P: reachable states, execution traces, etc.

Chapter #1:

Rules, Axioms, Trees

Inference Rule

A *proof system* is formed from a set of *inference rules* chained together to form proofs, also called *derivations*. Any derivation has only one final conclusion, which is the statement proved or derived. (Wiki)

$$\frac{\vdash a < b \quad \vdash b < c}{a < c}\,\text{R1}$$

Premises (known *facts*): $a < b$ and $b < c$.
(antecedent)

Conclusion (new fact): $a < c$.
(consequent)

Axiom

An *axiom* is an inference rule without a premise.

$$\frac{}{\vdash x \times 0 = 0}\, A_1$$

It reads: in any environment, the result of multiplication of x by zero equals to zero.

Transition Rule

A *transition rule* defines the conditions under which a system may be moved to a new *state*.

$$\frac{\langle \mathsf{a}, s\rangle \longrightarrow \langle n, s\rangle}{\langle \mathsf{a}{+}{+}, s\rangle \longrightarrow \langle n, s[\mathsf{a} \mapsto n+1]\rangle}$$

It reads: if a produces n without changing the state, then $\mathsf{a}{+}{+}$ may produce n changing the state by adding a new mapping $\mathsf{a} \mapsto n$.

The following set of transition rules may constitute the entire semantic of a language:

$$\frac{}{\langle \mathrm{x}, s\rangle \longrightarrow \langle s[\mathrm{x}], s\rangle} \qquad \frac{}{\langle n, s\rangle \longrightarrow \langle n, s\rangle}$$

$$\frac{\langle \mathrm{y}, s\rangle \longrightarrow \langle n, s\rangle}{\langle \mathrm{x} := \mathrm{y}, s\rangle \longrightarrow \langle \mathtt{skip}, s[\mathrm{x} \mapsto n]\rangle}$$

$$\frac{\langle C_1, s\rangle \longrightarrow \langle \mathtt{skip}, s'\rangle \quad \langle C_2, s'\rangle \longrightarrow \langle n, s''\rangle}{\langle C_1; C_2, s\rangle \longrightarrow \langle n, s''\rangle}$$

$$\frac{\langle \mathrm{x}, s\rangle \longrightarrow \langle n, s\rangle}{\langle \mathrm{x}{+}{+}, s\rangle \longrightarrow \langle n, s[\mathrm{x} \mapsto n+1]\rangle}$$

Proof Tree

Let's prove that `a:=5;a++++;` equals to 6:

Transition rules:

$$\frac{}{\langle \mathtt{x}, s\rangle \longrightarrow \langle s[\mathtt{x}], s\rangle} \qquad \frac{}{\langle n, s\rangle \longrightarrow \langle n, s\rangle}$$

$$\frac{\langle \mathtt{y}, s\rangle \longrightarrow \langle n, s\rangle}{\langle \mathtt{x:=y}, s\rangle \longrightarrow \langle \mathtt{skip}, s[\mathtt{x} \mapsto n]\rangle}$$

$$\frac{\langle C_1, s\rangle \longrightarrow \langle \mathtt{skip}, s'\rangle \qquad \langle C_2, s'\rangle \longrightarrow \langle n, s''\rangle}{\langle C_1; C_2, s\rangle \longrightarrow \langle n, s''\rangle}$$

$$\frac{\langle \mathtt{x}, s\rangle \longrightarrow \langle n, s\rangle}{\langle \mathtt{x{+}{+}}, s\rangle \longrightarrow \langle n, s[\mathtt{x} \mapsto n+1]\rangle}$$

Proof tree:

$$\frac{\dfrac{\dfrac{}{\langle 5, \{\}\rangle \longrightarrow \langle 5, \{\}\rangle}}{\langle \mathtt{a:=5}, \{\}\rangle \longrightarrow \langle \mathtt{skip}, \{\mathtt{a} \mapsto 5\}\rangle} \qquad \dfrac{\dfrac{\dfrac{}{\langle \mathtt{a}, \{\mathtt{a} \mapsto 5\}\rangle \longrightarrow \langle 5, \{\mathtt{a} \mapsto 5\}\rangle}}{\langle \mathtt{a{+}{+}}, \{\mathtt{a} \mapsto 5\}\rangle \longrightarrow \langle 5, \{\mathtt{a} \mapsto 6\}\rangle}}{\langle \mathtt{a{+}{+}{+}{+}}, \{\mathtt{a} \mapsto 5\}\rangle \longrightarrow \langle 6, \{\mathtt{a} \mapsto 7\}\rangle}}{\langle \mathtt{a:=5;a{+}{+}{+}{+}}, \{\}\rangle \longrightarrow \langle 6, \{\mathtt{a} \mapsto 7\}\rangle}$$

Chapter #2:

Operational vs Denotational

The *denotational semantics* assign to every expression the *number* denoted by that expression:

$$\Downarrow \subseteq \mathcal{A} \times \mathcal{D}$$
$$x^n \Downarrow y \quad \text{where } y = x \times x \times x \cdots \times x \quad (n \text{ times})$$

The *operational semantics* describe the computation steps taken in order to evaluate the expression to *normal form*:

$$\leadsto \subseteq \mathcal{A} \times \mathcal{A}$$
$$1)\ x^n \leadsto x \times x^{n-1} \quad \text{if } x > 0 \qquad 2)\ x^0 \leadsto 1$$

The operational semantics is the specification of an *interpreter* for the programming language whereas the denotational semantics tries to capture the *mathematical essence* of a program, abstracting away from computational details.

Chapter #3:

Natural Semantic (Denotational)

Syntax: `FASTER; STOP; SLOWER;`
Semantic ($\Downarrow \subseteq \langle \texttt{COMMAND}, \mathbb{N} \rangle \times \langle \mathbb{B}, \mathbb{N} \rangle$):

$$\frac{}{\langle \texttt{STOP}, s \rangle \Downarrow \langle \mathbf{tt}, 0 \rangle}\text{R1}$$

$$\frac{s < 60}{\langle \texttt{FASTER}, s \rangle \Downarrow \langle \mathbf{tt}, s + 20 \rangle}\text{R2} \qquad \frac{s \geq 60}{\langle \texttt{FASTER}, s \rangle \Downarrow \langle \mathbf{ff}, s \rangle}\text{R3}$$

$$\frac{}{\langle \texttt{SLOWER}, s \rangle \Downarrow \langle \mathbf{tt}, \mathsf{max}(0, s - 20) \rangle}\text{R4}$$

$$\frac{\langle C_1, s \rangle \Downarrow \langle r_1, s' \rangle \quad \langle C_2, s' \rangle \Downarrow \langle r_2, s'' \rangle}{\langle C_1; C_2, s \rangle \Downarrow \langle r_1 \wedge r_2, s'' \rangle}\text{R5}$$

Introduced by Gilles Kahn in 1987.

Proof Tree

$$
\dfrac{
  \dfrac{\dfrac{45 < 60}{\langle \texttt{FASTER}, 45\rangle \Downarrow \langle \mathbf{tt}, 65\rangle}\,\text{R2}
  \quad
  \dfrac{65 > 60}{\langle \texttt{FASTER}, 65\rangle \Downarrow \langle \mathbf{ff}, 65\rangle}\,\text{R3}}
  {\langle \texttt{FASTER}; \texttt{FASTER}, 45\rangle \Downarrow \langle \mathbf{ff}, 65\rangle}\,\text{R5}
  \qquad
  \langle \texttt{SLOWER}, 65\rangle \Downarrow \langle \mathbf{tt}, 45\rangle
}{\langle \texttt{FASTER}; \texttt{FASTER}; \texttt{SLOWER}, 45\rangle \Downarrow \langle \mathbf{ff}, 45\rangle}\,\text{R5}
$$

The tree is built from the bottom to the top, using the rules introduced before. The gray conditions at the top are not parts of the rules, that's why in gray.

Chapter #4:

Structural Semantic (Operational)

Consider a program:

```
x := x + 1;
```

The meaning of it may be explained by the SOS rule:

$$\frac{\langle e, s\rangle \longrightarrow \langle n, s\rangle}{\langle a := e, s\rangle \longrightarrow \langle \texttt{skip}, s[a \mapsto n]\rangle}$$

It reads: If e may be *evaluated* to n, then $a := e$ inserts a new binding $a \mapsto n$ to the state, and skips any further processing. To understand the meaning of `x+1` a new SOS rule is required.

Introduced by Gordon Plotkin in 1981.

Chapter #5:

Reduction Semantic

Consider a λ-*expression*:

$$(\lambda a.a)b$$

In Java it would look like this:

```
int f(int a) { return a; }
int x = f(b);
```

The expression may be reduced using so called β-*reduction*:

$$(\lambda x.t)s \longrightarrow t[x := s]$$

Thus

$$(\lambda a.a)b \quad \longrightarrow \quad b$$

Normal Form

A *normal form* is a form that has no more possible applications of reductions. This not a normal form:

$$(\lambda a.a)((\lambda b.b)((\lambda c.c)d))$$

It may be reduced to a normal form:

$$\begin{aligned} &\longrightarrow_\beta \quad (\lambda a.a)((\lambda b.b)d) \\ &\longrightarrow_\beta \quad (\lambda a.a)d \\ &\longrightarrow_\beta \quad d \end{aligned}$$

No further reductions are possible.

Chapter #6:

Further Reading/Watching

Christopher Strachey (2000),
Fundamental Concepts in Programming Languages

Alexander Kurz (2022),
Operational and Denotational Semantics

Michael Pradel (2021),
Lectures on "Operational Semantics"

Gordon Plotkin (1981),
A Structural Approach to Operational Semantics

References