Interpreting Big-Step Semantics in Small-Step Style

David A. Schmidt
Kansas State University*

Kahn-style ("big-step") semantics [2] is convenient because an entire computation is captured within a (semi)compositional derivation that can be reasoned about inductively. Although one cannot use Kahn-style semantics to reason about divergent programs, it is common to implement a big-step semantics as a set of Prolog Horn clauses and incrementally develop a derivation from the derivation's root to its leaves; if the computation is divergent, an infinite tree results [5].

We attempt to formalize the intuitions within the Prolog implementation of big-step semantics, and in doing so, expose the relationship between big-step and small step derivations. Several stages of development are needed.

We deal with *L-attributed* Kahn-style rules: A rule of the form

$$\frac{\rho_1 \vdash e_1 \Downarrow v_1 \quad \cdots \quad \rho_n \vdash e_n \Downarrow v_n}{\rho_0 \vdash op(e_1, ..., e_n) \Downarrow v_{n+1}}$$

is L-attributed if the value of each ρ_i is dependent upon only those ρ_j and v_j such that $j < i$ and if the value of each v_i is dependent only upon those ρ_j such that $j \leq i$ and v_j such that $j < i$. The intuition is that a derivation with an L-attributed rule can be formulated with a left-to-right tree traversal.

Next, we interpret the rule scheme as inducing a family of *positive* and *negative* rules, in the sense of Cousot and Cousot [1, 4]: The rule scheme induces a family of *positive* rules of the form

$$+ : \frac{\rho_1 \vdash e_1 \Downarrow v_1 \quad \cdots \quad \rho_n \vdash e_n \Downarrow v_n}{\rho_0 \vdash op(e_1, ..., e_n) \Downarrow v_{n+1}}$$

and n families of *negative* rules of the form

$$- : \frac{\rho_1 \vdash e_1 \Downarrow v_1 \cdots \rho_i \vdash e_i \Downarrow v_i \quad \rho_{i+1} \vdash e_{i+1} \Uparrow}{\rho_0 \vdash op(e_1, ..., e_n) \Uparrow}$$

The positive rules derive convergent computations and the negative rules derive divergent ones. When one first takes an inductive (least-fixed point) interpretation of the positive rules and then takes a coinductive (greatest fixed point) interpretation of the negative rules, the result is the set of well-formed derivations for programs in the source programming language. With the inductive interpretation come inductive reasoning principles, most notably induction on the height of the derivations, and with the coinductive interpretation comes coinduction.

When an initial environment, program pair, ρ_0, e_0, are used to initiate a derivation, the derivor has no clue whether a convergent derivation, $\rho_0 \vdash e_0 \Downarrow v$, or a divergent derivation, $\rho_0 \vdash e_0 \Uparrow$, will develop. A Prolog-style "logical variable" is often used to represent this lack of knowledge, but a formalization should employ other means. Based on the intuition that the triple, $\rho_0 \vdash e_0 \Downarrow v$, asserts the existence of a finite derivation of small steps, $(\rho_0 \vdash e_0) \rightarrow^* v$, we propose this new grammar for derivations:

$$\begin{aligned} & d \in \mathit{Derivation} \\ & s \in \mathit{Sequent}(\rho \vdash e) \\ & v \in \mathit{Value} \\ d \ ::= \ & v \ \mid \ s(d_1, \ldots, d_n), \ n \geq 0 \end{aligned}$$

That is, a derivation is a value, e.g., an integer, or a tree whose root is a sequent, $\rho_0 \vdash e_0$,

*Computing and Information Sciences Department, 234 Nichols Hall, Manhattan, KS 66506 USA. Ph: +1-785-532-6350. schmidt@cis.ksu.edu. Supported by NSF CCR-9633388.

and whose subtrees are derivations. If a sequent, s, has a sequence of computation steps that converge at v, then the rightmost subtree for the derivation whose root is s will be a leaf, v. Here's a small example:

$$
\begin{tikzcd}
 & & \rho \vdash (\lambda x.x)(1+2) \arrow[dll] \arrow[d] \arrow[drr] \arrow[drrr] & & & \\
\rho \vdash (\lambda x.x) \arrow[d] & & \rho \vdash 1+2 \arrow[dl] \arrow[d] \arrow[dr] & & \rho_3 \vdash x \arrow[d] & 3 \\
\langle \rho, x, x \rangle & \rho \vdash 1 \arrow[d] & \rho \vdash 2 \arrow[d] & 3 & 3 & \\
 & 1 & 2 & & &
\end{tikzcd}
$$

Such trees bear a relationship to the ones derived with Plotkin-style ("small-step") rules [3], when one notes that some of the above arrows represent "congruence" steps whereas others represent "$\beta\delta$" steps.

The following auxiliary definitions are useful:

$converge : Derivation \to Bool$
$\quad converge(d)$ iff $d \in Value$
$\quad\quad$ or $d = s(d_1, \ldots, d_n)$ and $converge(d_n)$
$\quad\quad$ (take lfp interpretation)
$diverge : Derivation \to Bool$
$\quad diverge(d)$ iff $d = s(d_1, \ldots, d_n)$
$\quad\quad$ and $diverge(d_n)$
$\quad\quad$ (take gfp interpretation)
$valueof : Derivation \to Value$
$\quad valueof(d) = v,$ where $d = s(d_1, \ldots, v)$
$\quad\quad$ (defined only when $converge(d)$ holds)
$root : Derivation \to (Value \cup Sequent)$
$\quad root(v) = v$
$\quad root(s(d_1, \ldots, d_n)) = s$

With these definitions, one can recode a positive inference rule,

$$+ : \frac{\rho_1 \vdash e_1 \Downarrow v_1 \quad \cdots \quad \rho_n \vdash e_n \Downarrow v_n}{\rho_0 \vdash op(e_1, \ldots, e_n) \Downarrow v_{n+1}}$$

into the new syntax:

$s(d_1, \ldots, d_n),$
$\quad$ where $\quad s = \rho_0 \vdash op(e_1, \ldots, e_n),$
$\quad root(d_i) = \rho_i \vdash e_i, 1 \le i \le n-1,$
$\quad converge(d_i)$ and $valueof(d_i) = v_i,$
$\quad\quad$ for all $1 \le i \le n-1,$
$\quad$ and $d_n = v_n$

The format for the negative rules is similar.

With a number of straightforward proofs, one can show that the set of derivations in the new syntax is isomorphic to the set of derivations in the original syntax.

The next major step is justifying why an incremental, top-down proof search creates a well-formed derivation. With the new syntax, one can prove *soundness*: A top-down, left-to-right, incrementally developed tree must be a well-formed derivation. Further, there is an *adequacy* result: For L-attributed Kahn-style rules, every well-formed derivation can be discovered by an incremental, top-down derivation.

Of course, one might argue that the incremental derivation is itself a small-step semantics. Perhaps this is so, but a formal connection between an incremental derivation and the one performed by a "congruent" Plotkin-style semantics is desirable, primarily because the congruence might enable one to prove properties of one formulation that carry over to the other.

References

[1] P. Cousot and R. Cousot. Inductive definitions, semantics, and abstract interpretation. In *Proc. 19th ACM Symp. on Principles of Programming Languages*, pages 83–94. ACM Press, 1992.

[2] G. Kahn. Natural semantics. In *Proc. STACS '87*, pages 22–39. Lecture Notes in Computer Science 247, Springer, Berlin, 1987.

[3] Gordon D. Plotkin. A structural approach to operational semantics. Technical Report FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark, September 1981.

[4] D.A. Schmidt. Trace-based abstract interpretation in the operational semantics hierarchy. Technical Report CIS-97-6, `http://www.cis.ksu.edu/~schmidt/papers/aiosh.ps.Z`, Kansas State University, Manhattan, Kansas, 1997.

[5] A. Stoughton. Treating finite and infinite operational semantics derivation trees as objects. Technical Report 97-12, Computing Science, Kansas State University, 1997.