Chapter 9

IMP: An Imperative Programming Language

Up to now, you have been exposed to some simple, expression-oriented programming languages, which are sufficient to demonstrate the basics of techniques for developing semantics, but seem a far cry from most real-world programming languages. These notes introduce a language that probably has more in common with a typical *imperative* programming language. The IMP language is a classic model programming language used to teach semantics. While not still rudimentary, IMP's syntax and semantics is substantially more complex than what you have seen so far. Working with that complexity should give you more experience with the techniques we have learned, and give you a taste of concepts that will be further developed later.

9.1 IMP By Example

To start off, let's consider a few IMP programs, and ponder their behaviour.

Syntax

IMP Language syntax is structured in three layers, arithmetic expressions, Boolean expressions, and commands. As we wil shortly see, the semantcs are layered similarly.

$$n \in \mathbb{Z}, \quad bv \in \textsc{Bool}, \quad X \in \textsc{Loc}, \quad a \in \textsc{AExp}, \quad b \in \textsc{BExp}, \quad c \in \textsc{Com},$$
$$\begin{array}{rcl} a & ::= & X \mid n \mid a + a \mid a - a \mid a * a \\ b & ::= & \mathsf{true} \mid \mathsf{false} \mid a = a \mid a \leq a \mid \neg b \mid b \wedge b \mid b \vee b \\ c & ::= & \mathsf{skip} \mid X := a \mid c; c \mid \mathsf{if}\ b\ \mathsf{then}\ c\ \mathsf{else}\ c \mid \mathsf{while}\ b\ \mathsf{do}\ c \\ bv & ::= & \mathsf{true} \mid \mathsf{false} \end{array}$$

Big-step Semantics

$$\sigma \in \textsc{Store} = \textsc{Loc} \to \mathbb{Z}$$
$$\textsc{ACfg} = \textsc{AExp} \times \textsc{Store}, \quad \textsc{BCfg} = \textsc{BExp} \times \textsc{Store}, \quad \textsc{CCfg} = \textsc{Com} \times \textsc{Store}$$

$$\sigma_z \in \textsc{Store}$$
$$\sigma_z(X) = 0$$

$$
\begin{aligned}
&\cdot[\cdot \mapsto \cdot] : \textsc{Store} \times \textsc{Loc} \times \mathbb{N} \to \textsc{Store} \\
&\sigma[X_0 \mapsto n](X_0) = n \\
&\sigma[X_0 \mapsto n](X_1) = \sigma(X_1) \quad \text{if } X_0 \neq X_1
\end{aligned}
$$

$\Downarrow_{\text{AExp}} \subseteq \text{ACfg} \times \mathbb{N}$

$$\frac{}{\langle n, \sigma\rangle \Downarrow_{\text{AExp}} n}\ (\text{enum}) \qquad \frac{}{\langle X, \sigma\rangle \Downarrow_{\text{AExp}} \sigma(X)}\ (\text{eloc}) \qquad \frac{\langle a_1, \sigma\rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 ++ a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_1 + n_2}\ (\text{eplus})$$

$$\frac{\langle a_1, \sigma\rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 \text{-} a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_1 - n_2}\ (\text{eminus}) \qquad \frac{\langle a_1, \sigma\rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 * a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_1 * n_2}\ (\text{etimes})$$

$\Downarrow_{\text{BExp}} \subseteq \text{BCfg} \times \text{Bool}$

$$\frac{}{\langle \mathsf{true}, \sigma\rangle \Downarrow_{\text{BExp}} \mathsf{true}}\ (\text{etrue}) \qquad \frac{}{\langle \mathsf{false}, \sigma\rangle \Downarrow_{\text{BExp}} \mathsf{false}}\ (\text{efalse})$$

$$\frac{\langle a_1, \sigma\rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 == a_2, \sigma\rangle \Downarrow_{\text{BExp}} bv}\ (\text{eeq}) \quad \begin{cases} bv = \mathsf{true} \text{ if } n_1 = n_2 \\ bv = \mathsf{false} \text{ if } n_1 \neq n_2 \end{cases}$$

$$\frac{\langle a_1, \sigma\rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 \leq a_2, \sigma\rangle \Downarrow_{\text{BExp}} bv}\ (\text{eleq}) \quad \begin{cases} bv = \mathsf{true} \text{ if } n_1 \leq n_2 \\ bv = \mathsf{false} \text{ if } n_1 > n_2 \end{cases}$$

$$\frac{\langle b, \sigma\rangle \Downarrow_{\text{BExp}} bv_1}{\langle \neg b, \sigma\rangle \Downarrow_{\text{BExp}} bv_2}\ (\text{enot}) \quad \begin{cases} bv_2 = \mathsf{true} \text{ if } bv_1 = \mathsf{false} \\ bv_2 = \mathsf{false} \text{ if } bv_1 = \mathsf{true} \end{cases}$$

$$\frac{\langle b_1, \sigma\rangle \Downarrow_{\text{BExp}} bv_1 \quad \langle b_2, \sigma\rangle \Downarrow_{\text{BExp}} bv_2}{\langle b_1 \wedge b_2, \sigma\rangle \Downarrow_{\text{BExp}} bv_3}\ (\text{eand}) \quad \begin{cases} bv_3 = \mathsf{true} \text{ if } bv_1 = bv_2 = \mathsf{true} \\ bv_3 = \mathsf{false} \text{ if otherwise} \end{cases}$$

$$\frac{\langle b_1, \sigma\rangle \Downarrow_{\text{BExp}} bv_1 \quad \langle b_2, \sigma\rangle \Downarrow_{\text{BExp}} bv_2}{\langle b_1 \vee b_2, \sigma\rangle \Downarrow_{\text{BExp}} bv_3}\ (\text{eor}) \quad \begin{cases} bv_3 = \mathsf{true} \text{ if } bv_1 = \mathsf{true} \text{ or } bv_2 = \mathsf{true} \\ bv_3 = \mathsf{false} \text{ if otherwise} \end{cases}$$

$\Downarrow_{\text{Com}} \subseteq \text{CCfg} \times \text{Store}$

$$\frac{}{\langle \mathsf{skip}, \sigma\rangle \Downarrow_{\text{Com}} \sigma}\ (\text{eskip}) \qquad \frac{\langle a, \sigma\rangle \Downarrow_{\text{AExp}} n}{\langle X ::== a, \sigma\rangle \Downarrow_{\text{Com}} \sigma[X \mapsto n]}\ (\text{eassign})$$

$$\frac{\langle c_1, \sigma\rangle \Downarrow_{\text{Com}} \sigma' \quad \langle c_2, \sigma'\rangle \Downarrow_{\text{Com}} \sigma''}{\langle c_1;; c_2, \sigma\rangle \Downarrow_{\text{Com}} \sigma''}\ (\text{eseq}) \qquad \frac{\langle b, \sigma\rangle \Downarrow_{\text{BExp}} \mathsf{true} \quad \langle c_1, \sigma\rangle \Downarrow_{\text{Com}} \sigma'}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \sigma\rangle \Downarrow_{\text{Com}} \sigma'}\ (\text{eif-t})$$

$$\frac{\langle b, \sigma\rangle \Downarrow_{\text{BExp}} \mathsf{false} \quad \langle c_2, \sigma\rangle \Downarrow_{\text{Com}} \sigma'}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \sigma\rangle \Downarrow_{\text{Com}} \sigma'}\ (\text{eif-f}) \qquad \frac{\langle b, \sigma\rangle \Downarrow_{\text{BExp}} \mathsf{false}}{\langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma\rangle \Downarrow_{\text{Com}} \sigma}\ (\text{ewhile-f})$$

$$\frac{\langle b, \sigma\rangle \Downarrow_{\text{BExp}} \mathsf{true} \quad \langle c, \sigma\rangle \Downarrow_{\text{Com}} \sigma' \quad \langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma'\rangle \Downarrow_{\text{Com}} \sigma''}{\langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma\rangle \Downarrow_{\text{Com}} \sigma''}\ (\text{ewhile-t})$$

$$\text{Pgm} = \text{Com}, \quad \text{Obs} = \text{Store} \cup \{\infty\}$$

$eval_{IMP} : \text{Pgm} \to \text{Obs}$
$eval_{IMP}(c) = \sigma \text{ if } \langle c, \sigma_z\rangle \Downarrow_{\text{Com}} \sigma$
$eval_{IMP}(c) = \infty \text{ otherwise}$

[RG: Stuffing in here a different, newer, IMP writeup. Merge these two!!]

9.2 IMPYIMPYIMP!

Up to now, you have been exposed to some simple, expression-oriented programming languages, which are sufficient to demonstrate the basics of techniques for developing semantics, but seem a far cry from most real-world programming languages. These notes introduce a language that probably has more in common with a typical *imperative* programming language. The IMP language is a classic model programming language used to teach semantics. While not still rudimentary, IMP's syntax and semantics is substantially more complex than what you have seen so far. Working with that complexity should give you more experience with the techniques we have learned, and give you a taste of concepts that will be further developed later.

9.3 IMP By Example

To start off, let's consider a few IMP programs, and ponder their behaviour.

Syntax

IMP Language syntax is structured in three layers, arithmetic expressions, Boolean expressions, and commands. As we wil shortly see, the semantcs are layered similarly.

$$
\begin{array}{lcl}
n \in \mathbb{Z}, & bv \in \textsc{Bool}, & X \in \textsc{Loc}, \quad a \in \textsc{AExp}, \quad b \in \textsc{BExp}, \quad c \in \textsc{Com}, \\
a & ::= & X \mid n \mid a + a \mid a - a \mid a * a \\
b & ::= & \mathsf{true} \mid \mathsf{false} \mid a = a \mid a \leq a \mid \neg b \mid b \wedge b \mid b \vee b \\
c & ::= & \mathsf{skip} \mid X := a \mid c; c \mid \mathsf{if}\ b\ \mathsf{then}\ c\ \mathsf{else}\ c \mid \mathsf{while}\ b\ \mathsf{do}\ c \\
bv & ::= & \mathsf{true} \mid \mathsf{false}
\end{array}
$$

Big-step Semantics

$$\sigma \in \textsc{Store} = \textsc{Loc} \to \mathbb{Z}$$
$$\textsc{ACfg} = \textsc{AExp} \times \textsc{Store}, \quad \textsc{BCfg} = \textsc{BExp} \times \textsc{Store}, \quad \textsc{CCfg} = \textsc{Com} \times \textsc{Store}$$

$$\sigma_z \in \textsc{Store}$$
$$\sigma_z(X) = 0$$

$$
\begin{aligned}
&\cdot[\cdot \mapsto \cdot] : \textsc{Store} \times \textsc{Loc} \times \mathbb{N} \to \textsc{Store} \\
&\sigma[X_0 \mapsto n](X_0) = n \\
&\sigma[X_0 \mapsto n](X_1) = \sigma(X_1) \quad \text{if } X_0 \neq X_1
\end{aligned}
$$

$$\boxed{\Downarrow_{\text{AExp}} \ \subseteq \text{ACfg} \times \mathbb{N}}$$

$$\frac{}{\langle n, \sigma \rangle \Downarrow_{\text{AExp}} n}\ (\text{enum}) \qquad \frac{}{\langle X, \sigma \rangle \Downarrow_{\text{AExp}} \sigma(X)}\ (\text{eloc}) \qquad \frac{\langle a_1, \sigma \rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma \rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 + +a_2, \sigma \rangle \Downarrow_{\text{AExp}} n_1 + n_2}\ (\text{eplus})$$

$$\frac{\langle a_1, \sigma \rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma \rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 \text{-} a_2, \sigma \rangle \Downarrow_{\text{AExp}} n_1 - n_2}\ (\text{eminus}) \qquad \frac{\langle a_1, \sigma \rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma \rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 * a_2, \sigma \rangle \Downarrow_{\text{AExp}} n_1 * n_2}\ (\text{etimes})$$

$$\boxed{\Downarrow_{\text{BExp}} \ \subseteq \text{BCfg} \times \text{Bool}}$$

$$\frac{}{\langle \mathsf{true}, \sigma \rangle \Downarrow_{\text{BExp}} \mathsf{true}}\ (\text{etrue}) \qquad \frac{}{\langle \mathsf{false}, \sigma \rangle \Downarrow_{\text{BExp}} \mathsf{false}}\ (\text{efalse})$$

$$\frac{\langle a_1, \sigma \rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma \rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 == a_2, \sigma \rangle \Downarrow_{\text{BExp}} bv}\ (\text{eeq}) \quad \begin{cases} bv = \mathsf{true} \text{ if } n_1 = n_2 \\ bv = \mathsf{false} \text{ if } n_1 \neq n_2 \end{cases}$$

$$\frac{\langle a_1, \sigma \rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma \rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 \leq a_2, \sigma \rangle \Downarrow_{\text{BExp}} bv}\ (\text{eleq}) \quad \begin{cases} bv = \mathsf{true} \text{ if } n_1 \leq n_2 \\ bv = \mathsf{false} \text{ if } n_1 > n_2 \end{cases}$$

$$\frac{\langle b, \sigma \rangle \Downarrow_{\text{BExp}} bv_1}{\langle \neg b, \sigma \rangle \Downarrow_{\text{BExp}} bv_2}\ (\text{enot}) \quad \begin{cases} bv_2 = \mathsf{true} \text{ if } bv_1 = \mathsf{false} \\ bv_2 = \mathsf{false} \text{ if } bv_1 = \mathsf{true} \end{cases}$$

$$\frac{\langle b_1, \sigma \rangle \Downarrow_{\text{BExp}} bv_1 \quad \langle b_2, \sigma \rangle \Downarrow_{\text{BExp}} bv_2}{\langle b_1 \wedge b_2, \sigma \rangle \Downarrow_{\text{BExp}} bv_3}\ (\text{eand}) \quad \begin{cases} bv_3 = \mathsf{true} \text{ if } bv_1 = bv_2 = \mathsf{true} \\ bv_3 = \mathsf{false} \text{ if otherwise} \end{cases}$$

$$\frac{\langle b_1, \sigma \rangle \Downarrow_{\text{BExp}} bv_1 \quad \langle b_2, \sigma \rangle \Downarrow_{\text{BExp}} bv_2}{\langle b_1 \vee b_2, \sigma \rangle \Downarrow_{\text{BExp}} bv_3}\ (\text{eor}) \quad \begin{cases} bv_3 = \mathsf{true} \text{ if } bv_1 = \mathsf{true} \text{ or } bv_2 = \mathsf{true} \\ bv_3 = \mathsf{false} \text{ if otherwise} \end{cases}$$

$$\boxed{\Downarrow_{\text{Com}} \ \subseteq \text{CCfg} \times \text{Store}}$$

$$\frac{}{\langle \mathsf{skip}, \sigma \rangle \Downarrow_{\text{Com}} \sigma}\ (\text{eskip}) \qquad \frac{\langle a, \sigma \rangle \Downarrow_{\text{AExp}} n}{\langle X ::== a, \sigma \rangle \Downarrow_{\text{Com}} \sigma[X \mapsto n]}\ (\text{eassign})$$

$$\frac{\langle c_1, \sigma \rangle \Downarrow_{\text{Com}} \sigma' \quad \langle c_2, \sigma' \rangle \Downarrow_{\text{Com}} \sigma''}{\langle c_1 ; ; c_2, \sigma \rangle \Downarrow_{\text{Com}} \sigma''}\ (\text{eseq}) \qquad \frac{\langle b, \sigma \rangle \Downarrow_{\text{BExp}} \mathsf{true} \quad \langle c_1, \sigma \rangle \Downarrow_{\text{Com}} \sigma'}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \sigma \rangle \Downarrow_{\text{Com}} \sigma'}\ (\text{eif-t})$$

$$\frac{\langle b, \sigma \rangle \Downarrow_{\text{BExp}} \mathsf{false} \quad \langle c_2, \sigma \rangle \Downarrow_{\text{Com}} \sigma'}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \sigma \rangle \Downarrow_{\text{Com}} \sigma'}\ (\text{eif-f}) \qquad \frac{\langle b, \sigma \rangle \Downarrow_{\text{BExp}} \mathsf{false}}{\langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma \rangle \Downarrow_{\text{Com}} \sigma}\ (\text{ewhile-f})$$

$$\frac{\langle b, \sigma \rangle \Downarrow_{\text{BExp}} \mathsf{true} \quad \langle c, \sigma \rangle \Downarrow_{\text{Com}} \sigma' \quad \langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma' \rangle \Downarrow_{\text{Com}} \sigma''}{\langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma \rangle \Downarrow_{\text{Com}} \sigma''}\ (\text{ewhile-t})$$

$$\text{Pgm} = \text{Com}, \quad \text{Obs} = \text{Store} \cup \{\infty\}$$

$$\begin{aligned} &eval_{IMP} : \text{Pgm} \to \text{Obs} \\ &eval_{IMP}(c) = \sigma \text{ if } \langle c, \sigma_z \rangle \Downarrow_{\text{Com}} \sigma \\ &eval_{IMP}(c) = \infty \text{ otherwise} \end{aligned}$$

Bibliography

P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, chapter C.7, pages 739–782. North-Holland, 1977.

J. Avigad. Reliability of mathematical inference. *Synthese*, 2020. doi: 10.1007/s11229-019-02524-y. `https://doi.org/10.1007/s11229-019-02524-y`.

F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, New York, NY, USA, 1998. ISBN 0-521-45520-0.

J. Bagaria. Set theory. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. The Metaphysics Research Lab, winter 2014 edition, 2014. URL `http://plato.stanford.edu/archives/win2014/entries/set-theory/`.

R. Baldoni, E. Coppa, D. C. D'elia, C. Demetrescu, and I. Finocchi. A survey of symbolic execution techniques. *ACM Comput. Surv.*, 51(3):50:1–50:39, May 2018. ISSN 0360-0300. doi: 10.1145/3182657. URL `http://doi.acm.org/10.1145/3182657`.

H. P. Barendregt. *The lambda calculus - its syntax and semantics*, volume 103 of *Studies in logic and the foundations of mathematics*. North-Holland, 1985. ISBN 978-0-444-86748-3.

A. Bauer. Proof of negation and proof by contradiction. Mathematics and Computation Blog, March 2010. `http://math.andrej.com/2010/03/29/proof-of-negation-and-proof-by-contradiction/`.

L. Crosilla. Set Theory: Constructive and Intuitionistic ZF. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2020 edition, 2020.

M. Felleisen and D. P. Friedman. A syntactic theory of sequential state. *Theoretical Computer Science*, 69(3):243–287, 1989. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(89)90069-8. URL `http://www.sciencedirect.com/science/article/pii/0304397589900698`.

M. Felleisen, R. B. Findler, and M. Flatt. *Semantics Engineering with PLT Redex*. MIT Press, 1st edition, 2009.

J. Ferreirós. The early development of set theory. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2019 edition, 2019. URL `https://plato.stanford.edu/archives/sum2019/entries/settheory-early/`.

D. Grossman, G. Morrisett, T. Jim, M. Hicks, Y. Wang, and J. Cheney. Region-based memory management in cyclone. In *Proceedings of the ACM SIGPLAN 2002 Conference on Programming Language Design and Implementation*, PLDI '02, pages 282–293, New York, NY, USA, 2002. ACM. ISBN 1-58113-463-0. doi: 10.1145/512529.512563. URL `http://doi.acm.org/10.1145/512529.512563`.

P. R. Halmos. *Naive Set Theory*. Springer-Verlag, first edition, Jan. 1960. ISBN 0387900926.
A classic introductory textbook on set theory.

P. R. Harper. *Practical Foundations for Programming Languages*. Cambridge University Press, New York, NY, USA, 2012. URL `http://www.cs.cmu.edu/%7Erwh/plbook/1sted-revised.pdf`.

D. J. Howe. Proving congruence of bisimulation in functional programming languages. *Inf. Comput.*, 124 (2):103–112, Feb. 1996. ISSN 0890-5401.

G. Kahn. Natural semantics. In *4th Annual Symposium on Theoretical Aspects of Computer Sciences on STACS 87*, pages 22–39, London, UK, UK, 1987. Springer-Verlag.

R. Krebbers. *The C standard formalized in Coq.* PhD thesis, Radboud University Nijmegen, December 2015. URL `https://robbertkrebbers.nl/thesis.html`.

C. Kuratowski. Sur la notion de l'ordre dans la théorie des ensembles. *Fundamenta Mathematicae*, 2(1): 161–171, 1921. doi: 10.4064/fm-2-1-161-171. `https://web.archive.org/web/20190429103938/http://matwbn.icm.edu.pl/ksiazki/fm/fm2/fm2122.pdf`.

P. J. Landin. The mechanical evaluation of expressions. *Comput. J.*, 6(4):308–320, 1964. doi: 10.1093/comjnl/6.4.308. URL `https://doi.org/10.1093/comjnl/6.4.308`.

X. Leroy and H. Grall. Coinductive big-step operational semantics. *Inf. Comput.*, 207(2):284–304, Feb. 2009. ISSN 0890-5401.

P. Maddy. Believing the axioms. I. *The Journal of Symbolic Logic*, 53(02):481–511, 1988.
An interesting (though complicated) analysis of why set theorists believe in their axioms.

J. McCarthy. Recursive functions of symbolic expressions and their computation by machine, part I. *Commun. ACM*, 3(4):184–195, 1960.

J. H. J. Morris. *Lambda-Calculus Models of Programming Languages.* PhD thesis, Massachusetts Institute of Technology, Feb. 1969. URL `http://hdl.handle.net/1721.1/64850`.

A. M. Pitts. *Nominal sets: Names and symmetry in computer science.* Cambridge University Press, 2013.

G. D. Plotkin. The origins of structural operational semantics. *J. Log. Algebr. Program.*, 60-61:3–15, 2004a. doi: 10.1016/j.jlap.2004.03.009. URL `http://dx.doi.org/10.1016/j.jlap.2004.03.009`.

G. D. Plotkin. A structural approach to operational semantics. *J. Log. Algebr. Program.*, 60-61:17–139, 2004b.

B. Popik. "pull yourself up by your bootstraps". Weblog entry, September 2012. `https://www.barrypopik.com/index.php/new_york_city/entry/pull_yourself_up_by_your_bootstraps/`.

E. Reck and G. Schiemer. Structuralism in the Philosophy of Mathematics. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Metaphysics Research Lab, Stanford University, Spring 2020 edition, 2020.

D. Sangiorgi. On the origins of bisimulation and coinduction. *ACM Trans. Program. Lang. Syst.*, 31(4): 15:1–15:41, May 2009. ISSN 0164-0925.

W. Sieg and D. Schlimm. Dedekind's analysis of number: Systems and axioms. *Synthese*, 147(1):121–170, Oct 2005.

J. Spolsky. The law of leaky abstractions. Joel on Software Blog, November 2002. `https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/`.

G. L. Steele. Debunking the "expensive procedure call"" myth or, procedure call implementations considered harmful or, lamdba: The ultimate goto. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1977. URL `http://dspace.mit.edu/handle/1721.1/5753`.

S. Stenlund. Descriptions in intuitionistic logic. In S. Kanger, editor, *Proceedings of the Third Scandinavian Logic Symposium*, volume 82 of *Studies in Logic and the Foundations of Mathematics*, pages 197 – 212. Elsevier, 1975. URL `http://www.sciencedirect.com/science/article/pii/S0049237X08707328`.

M. Tiles. Book Review: Stephen Pollard. Philosophical Introduction to Set Theory. *Notre Dame Journal of Formal Logic*, 32(1):161–166, 1990.
A brief introduction to the philosophical issues underlying set theory as a foundation for mathematics.

C. Urban and M. Norrish. A formal treatment of the Barendregt variable convention in rule inductions. In *Proceedings of the 3rd ACM SIGPLAN Workshop on Mechanized Reasoning about Languages with Variable Binding*, MERLIN '05, page 25–32, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595930728. doi: 10.1145/1088454.1088458. URL `https://doi.org/10.1145/1088454.1088458`.

C. Urban, S. Berghofer, and M. Norrish. Barendregt's variable convention in rule inductions. In *Proceedings of the 21st International Conference on Automated Deduction: Automated Deduction*, CADE-21, page 35–50, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 9783540735946. doi: 10.1007/978-3-540-73595-3_4. URL `https://doi.org/10.1007/978-3-540-73595-3_4`.

D. van Dalen. *Logic and structure (3. ed.)*. Universitext. Springer, 1994. ISBN 978-3-540-57839-0.

A. K. Wright and M. Felleisen. A syntactic approach to type soundness. *Inf. Comput.*, 115(1):38–94, Nov. 1994.

B. Zimmer. figurative "bootstraps". email to linguistlist mailing list, August 2005. `http://listserv.linguistlist.org/pipermail/ads-l/2005-August/052756.html`.