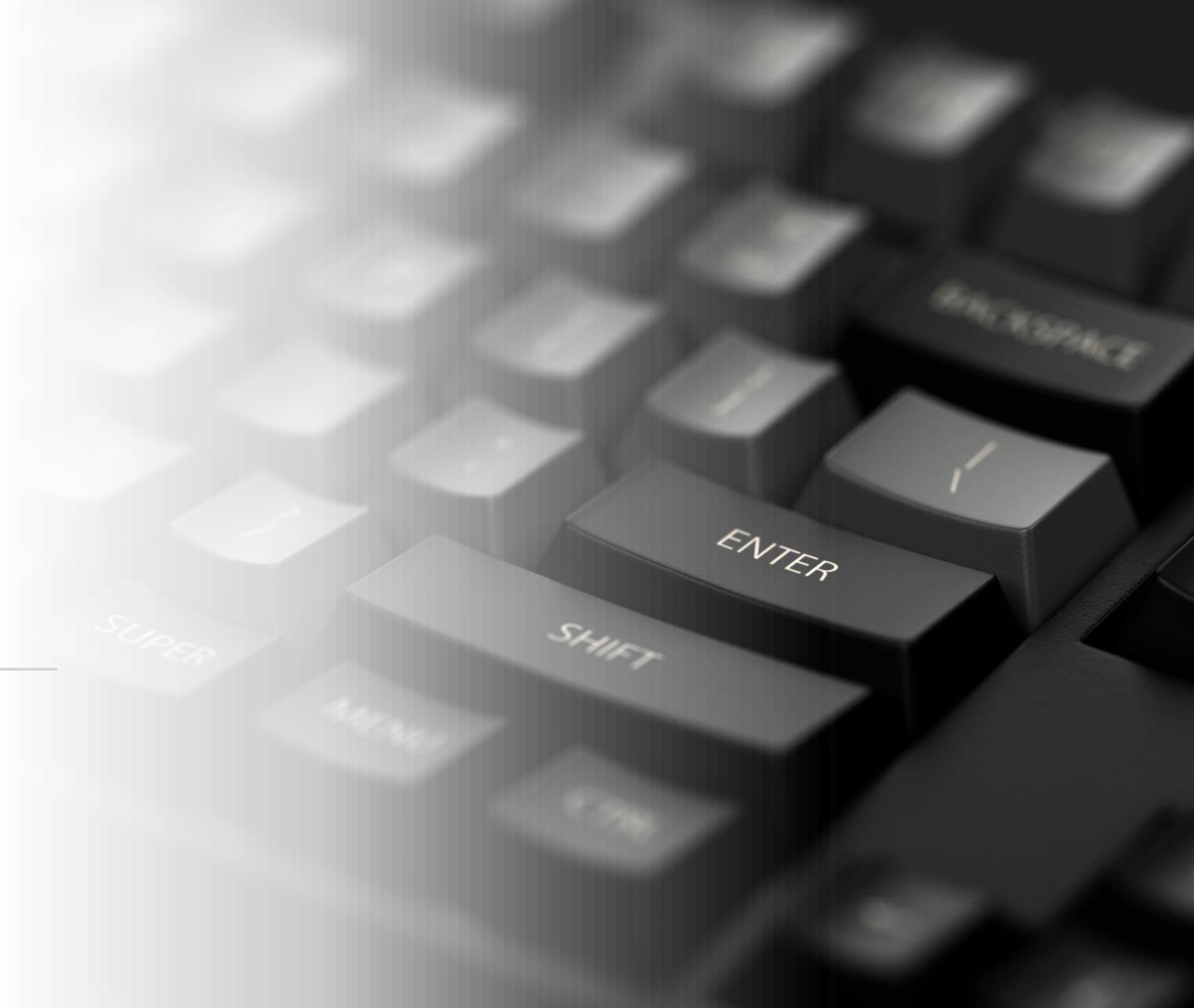
Programs



Algorithm

Program

Computation

A Simple, imperative language: IMP

sorts:

$Int,\ Bool,\ Id,\ AExp,\ BExp,\ Block,\ Stmt,\ Pgm$

subsorts:

$$
\begin{array}{rcl}
Int,\ Id & < & AExp \\
Bool & < & BExp \\
Block & < & Stmt
\end{array}
$$

operations:

$$
\begin{array}{rcl}
\_ + \_ & : & AExp \times AExp \to AExp \\
\_ / \_ & : & AExp \times AExp \to AExp \\
\_ <= \_ & : & AExp \times AExp \to BExp \\
!\, \_ & : & BExp \to BExp \\
\_ \,\&\&\, \_ & : & BExp \times BExp \to BExp \\
\{\} & : & \to Block \\
\{ \_ \} & : & Stmt \to Block \\
\_ = \_; & : & Id \times AExp \to Stmt \\
\_\ \_ & : & Stmt \times Stmt \to Stmt \\
\texttt{if}\,(\_)\,\_\,\texttt{else}\,\_ & : & BExp \times Block \times Block \to Stmt \\
\texttt{while}\,(\_)\,\_ & : & BExp \times Block \to Stmt \\
\texttt{int}\,\_;\,\_ & : & \mathbf{List}\{Id\} \times Stmt \to Pgm
\end{array}
$$

G. Winskel. The Formal Semantics of Programming Languages: An Introduction. MIT Press, 1993.

Big-Step Operational Semantics

A program state is a partial finite-domain function from identifiers to integers:

$\sigma : Id \rightharpoonup Int$

A configuration is a pair of a semantic ingredient and a program state:

$\langle a, \sigma \rangle$ or $\langle b, \sigma \rangle$ or $\langle s, \sigma \rangle$

Big Step Semantics have a kind of judgments:

$\langle configuration \rangle \Downarrow result$

Big-Step OS: Arithmetic

$$\langle i, \sigma \rangle \Downarrow \langle i \rangle \qquad \text{(BigStep-Int)}$$

$$\langle x, \sigma \rangle \Downarrow \langle \sigma(x) \rangle \quad \text{if } \sigma(x) \neq \bot \qquad \text{(BigStep-Lookup)}$$

$$\frac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 + a_2, \sigma \rangle \Downarrow \langle i_1 +_{Int} i_2 \rangle} \qquad \text{(BigStep-Add)}$$

$$\frac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 \mathbin{/} a_2, \sigma \rangle \Downarrow \langle i_1 \mathbin{/_{Int}} i_2 \rangle} \quad \text{if } i_2 \neq 0 \qquad \text{(BigStep-Div)}$$

Big-Step OS: Boolean

$$\langle t, \sigma \rangle \Downarrow \langle t \rangle \qquad \text{(BigStep-Bool)}$$

$$\frac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \quad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 \texttt{ <= } a_2, \sigma \rangle \Downarrow \langle i_1 \leq_{Int} i_2 \rangle} \qquad \text{(BigStep-Leq)}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \langle \texttt{true} \rangle}{\langle \texttt{!}\, b, \sigma \rangle \Downarrow \langle \texttt{false} \rangle} \qquad \text{(BigStep-Not-True)}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \langle \texttt{false} \rangle}{\langle \texttt{!}\, b, \sigma \rangle \Downarrow \langle \texttt{true} \rangle} \qquad \text{(BigStep-Not-False)}$$

$$\frac{\langle b_1, \sigma \rangle \Downarrow \langle \texttt{false} \rangle}{\langle b_1 \texttt{ \&\& } b_2, \sigma \rangle \Downarrow \langle \texttt{false} \rangle} \qquad \text{(BigStep-And-False)}$$

$$\frac{\langle b_1, \sigma \rangle \Downarrow \langle \texttt{true} \rangle \quad \langle b_2, \sigma \rangle \Downarrow \langle t \rangle}{\langle b_1 \texttt{ \&\& } b_2, \sigma \rangle \Downarrow \langle t \rangle} \qquad \text{(BigStep-And-True)}$$

Big-Step OS: Statement

$$\langle \{\}, \sigma\rangle \Downarrow \langle \sigma\rangle \qquad \text{(BigStep-Empty-Block)}$$

$$\frac{\langle s, \sigma\rangle \Downarrow \langle \sigma'\rangle}{\langle \{\ s\ \}, \sigma\rangle \Downarrow \langle \sigma'\rangle} \qquad \text{(BigStep-Block)}$$

$$\frac{\langle a, \sigma\rangle \Downarrow \langle i\rangle}{\langle x = a\texttt{;}, \sigma\rangle \Downarrow \langle \sigma[i/x]\rangle}\ \text{if } \sigma(x) \neq \bot \qquad \text{(BigStep-Asgn)}$$

$$\frac{\langle s_1, \sigma\rangle \Downarrow \langle \sigma_1\rangle \quad \langle s_2, \sigma_1\rangle \Downarrow \langle \sigma_2\rangle}{\langle s_1\ s_2, \sigma\rangle \Downarrow \langle \sigma_2\rangle} \qquad \text{(BigStep-Seq)}$$

$$\frac{\langle b, \sigma\rangle \Downarrow \langle \texttt{true}\rangle \quad \langle s_1, \sigma\rangle \Downarrow \langle \sigma_1\rangle}{\langle \texttt{if}\ (b)\ s_1\ \texttt{else}\ s_2, \sigma\rangle \Downarrow \langle \sigma_1\rangle} \qquad \text{(BigStep-If-True)}$$

$$\frac{\langle b, \sigma\rangle \Downarrow \langle \texttt{false}\rangle \quad \langle s_2, \sigma\rangle \Downarrow \langle \sigma_2\rangle}{\langle \texttt{if}\ (b)\ s_1\ \texttt{else}\ s_2, \sigma\rangle \Downarrow \langle \sigma_2\rangle} \qquad \text{(BigStep-If-False)}$$

$$\frac{\langle b, \sigma\rangle \Downarrow \langle \texttt{false}\rangle}{\langle \texttt{while}\ (b)\ s, \sigma\rangle \Downarrow \langle \sigma\rangle} \qquad \text{(BigStep-While-False)}$$

$$\frac{\langle b, \sigma\rangle \Downarrow \langle \texttt{true}\rangle \quad \langle s\ \texttt{while}\ (b)\ s, \sigma\rangle \Downarrow \langle \sigma'\rangle}{\langle \texttt{while}\ (b)\ s, \sigma\rangle \Downarrow \langle \sigma'\rangle} \qquad \text{(BigStep-While-True)}$$

Turing-completeness

IMP can simulate TM

- Store the finite control
- Store the non-blank portion of the tape
- Simulate the run of the TM

TM can simulate IMP

- Store the IMP code on the tape
- Store the program state on the tape
- TM states represent the program counter

Similar proof for any programming language!