Structural Induction
(the first draft)

Dmitry Boulytchev

March 14, 2018

We have considered two languages (a language of expressions $\mathscr{E}$ and a language of stack machine programs $\mathscr{P}$), and a compiler from the former to the latter. It can be formally proven, that the compiler is (fully) correct in the sense, given in the lecture 1. Due to the simplicity of the languages, the proof technique — *structural induction* — is simple as well.

First, we collect all needed definitions in one place (see Fig. 1). We simplified the description of stack machine semantics a little bit: first, we dropped off all instructions, which cannot be generated by the expression compiler, and then, we removed the input and output streams from the stack machine configurations, since they are never affected by the remaining instructions.

Lemma 1. *(Determinism) Let p be an arbitrary stack machine program, and let c, c_1 and c_2 be arbitrary configurations. Then*

$$c \xrightarrow{p} c_1 \wedge c \xrightarrow{p} c_2 \Rightarrow c_1 = c_2$$

Proof. Induction on the structure of p.

Base case. If $p = \varepsilon$, then, by the rule STOP_{SM}, $c_1 = c$ and $c_2 = c$. Since no other rule can be applied, we're done.

Induction step. If $p = \iota p'$, then, by condition, we have

$$\frac{c' \xrightarrow{p'} c_1}{c \xrightarrow{\iota p'} c_1}$$

and

$$\frac{c'' \xrightarrow{p'} c_2}{c \xrightarrow{\iota p'} c_2}$$

where c' and c'' depend only on c and ι. By the case analysis on ι we conclude, that $c' = c''$. Since p' is shorter, than p, we can apply the induction hypothesis, which gives us $c_1 = c_2$. □

$$
\begin{array}{rcll}
[\![n]\!] & = & \lambda\sigma.n & [\textsc{Const}] \\
[\![x]\!] & = & \lambda\sigma.\sigma x & [\textsc{Var}] \\
[\![A \otimes B]\!] & = & \lambda\sigma.([\![A]\!]\sigma \oplus [\![B]\!]\sigma) & [\textsc{Binop}]
\end{array}
$$

(a) Denotational semantics for expressions

$$
\frac{}{c \xrightarrow{\varepsilon} c} \quad [\textsc{Stop}_{SM}]
$$

$$
\frac{\langle (x \oplus y) :: st, s\rangle \xrightarrow{p} c'}{\langle y :: x :: st, s\rangle \xrightarrow{(\texttt{BINOP}\ \otimes)p} c'} \quad [\textsc{Binop}_{SM}]
$$

$$
\frac{\langle z :: st, s\rangle \xrightarrow{p} c'}{\langle st, s\rangle \xrightarrow{(\texttt{CONST}\ z)p} c'} \quad [\textsc{Const}_{SM}]
$$

$$
\frac{\langle (s\ x) :: st, s\rangle \xrightarrow{p} c'}{\langle st, s\rangle \xrightarrow{(\texttt{LD}\ x)p} c'} \quad [\textsc{LD}_{SM}]
$$

(b) Big-step operational semantics for stack machine

$$
\begin{array}{rcll}
[\![x]\!]^{\mathscr{E}}_{comp} & = & [\texttt{LD}\ x] & [\textsc{Var}_{comp}] \\
[\![n]\!]^{\mathscr{E}}_{comp} & = & [\texttt{CONST}\ n] & [\textsc{Const}_{comp}] \\
[\![A \otimes B]\!]^{\mathscr{E}}_{comp} & = & [\![A]\!]^{\mathscr{E}}_{comp} @ [\![B]\!]^{\mathscr{E}}_{comp} @ [\texttt{BINOP}\otimes]) & [\textsc{Binop}_{comp}]
\end{array}
$$

(c) Compilation

Figure 1: All relevant definitions

Lemma 2. *(Compositionality) Let $p = p_1 p_2$ be an arbitrary stack machine program, subdivided into arbitrary subprograms p_1 and p_2. Then,*

$$\forall c_1, c_2 : c_1 \xrightarrow{p} c_2 \Leftrightarrow \exists c' : c_1 \xrightarrow{p_1} c' \wedge c' \xrightarrow{p_2} c_2$$

Proof. Induction on the structure of p.

Base case. The base case $p = \varepsilon$ is trivial: use the rule STOP_{SM} and get $c' = c_2 = c_1$.

Induction step. When $p = \iota p'$, then there are two cases:

- Either $p_1 = \varepsilon$, then $c' = c_1$ trivially by the rule STOP_{SM}, and we're done.
- Otherwise $p_1 = \iota p_1'$, and, thus, $p = \iota p_1' p_2$. In order to prove the lemma, we need to prove two implications:

  1. Let $c_1 \xrightarrow{p=\iota p_1' p_2} c_2$. Technically, we need here to consider three cases (one for each type of the instruction ι), but in all cases the outcome would be the same: we have the picture

  $$\frac{c'' \xrightarrow{p_1' p_2} c_2}{c_1 \xrightarrow{p=\iota p_1' p_2} c_2}$$

  where c'' depends only on ι and c_1. Since $p_1' p_2$ is shorter, than p, we can apply the induction hypothesis, which gives us a configuration c', such, that $c'' \xrightarrow{p_1'} c'$ and $c' \xrightarrow{p_2} c_2$. The observation $c_1 \xrightarrow{\iota p_1'} c'$ concludes the proof (note, we implicitly use determinism here).

  2. Let there exists c', such that $c_1 \xrightarrow{\iota p_1'} c'$ and $c' \xrightarrow{p_2} c_2$. From the first relation we have

  $$\frac{c'' \xrightarrow{p_1'} c'}{c_1 \xrightarrow{\iota p_1'} c'}$$

  where c'' depends only on ι and c_1. Since $p_1' p_2$ is shorter, than p, we can apply the induction hypothesis, which gives us $c'' \xrightarrow{p_1' p_2} c_2$, and, thus, $c_1 \xrightarrow{\iota p_1' p_2} c_2$ (again, we implicitly use determinism here).

□

Theorem 1. *(Correctness of compilation) Let $e \in \mathscr{E}$ be arbitrary expression, s — arbitrary state, and st — arbitrary stack. Then*

$$\langle st, s \rangle \xrightarrow{[\![e]\!]^{\mathscr{E}}_{comp}} \langle ([\![e]\!]\, s) :: st, s \rangle \text{ iff } ([\![e]\!]\, s) \text{ is defined}$$

Proof. Induction on the structure of e.

Base case. There are two subcases:

1. e is a constant z. Then:

   - $[\![e]\!]\,s = z$ for each state s;
   - $[\![e]\!]^{\mathscr{E}}_{comp} = [\texttt{CONST z}]$;
   - $\langle st, s\rangle \xrightarrow{[\texttt{CONST z}]} \langle z :: st, s\rangle$ for arbitrary st and s.

   This concludes the first base case.

2. e is a variable x. Then:

   - $[\![s]\!]\,s = s\,x$ for each state s, such that $s\,x$ is defined;
   - $[\![e]\!]^{\mathscr{E}}_{comp} = [\texttt{LD x}]$;
   - $\langle st, s\rangle \xrightarrow{[\texttt{CONST z}]} \langle (s\,x) :: st, s\rangle$ for arbitrary st and arbitrary s, such that $s\,x$ is defined.

   This concludes the second base case.

Induction step. Let e be $A \otimes B$. Then:

- $[\![A \otimes B]\!]s = [\![A]\!]s \oplus [\![B]\!]s$ for each state s, such that both $[\![A]\!]s$ and $[\![B]\!]s$ are defined;
- $[\![A \otimes B]\!]^{\mathscr{E}}_{comp} = [\![A]\!]^{\mathscr{E}}_{comp} @ [\![B]\!]^{\mathscr{E}}_{comp} @ [\texttt{BINOP}\oplus]$;
- by the inductive hypothesis, for arbitrary st and s

$$\langle st, s\rangle \xrightarrow{[\![A]\!]^{\mathscr{E}}_{comp}} \langle ([\![A]\!]s) :: st, s\rangle \text{iff } ([\![A]\!]\,s) \text{ is defined}$$

and

$$\langle ([\![A]\!]s) :: st, s\rangle \xrightarrow{[\![B]\!]^{\mathscr{E}}_{comp}} \langle ([\![B]\!]s) :: ([\![A]\!]s) :: st, s\rangle \text{iff } ([\![A]\!]\,s) \text{ and } ([\![A]\!]\,s) \text{ are defined}$$

Taking into account the semantics of $\texttt{BINOP}\otimes$ and applying the compositionality lemma, the theorem follows.

□