Some abstract machines

Arithmetic expression

What we present is a very simplified version of the fundamental paper of McCarthy and Painter on correctness of a compiler for arithmetic expressions (1967). The expressions are

$$e \ ::= \ \mathsf{const}\ n \mid \mathsf{add}\ e\ e$$

and the semantics is

$$[\![\mathsf{const}\ n]\!] = n \qquad [\![\mathsf{add}\ e_0\ e_1]\!] = [\![e_0]\!] + [\![e_1]\!]$$

We define the instruction list (code) as

$$cd \ ::= \ \mathsf{LOAD}\ n\ cd \mid \mathsf{ADD}\ cd \mid \mathsf{HALT}$$

and the compilation function is

$$comp\ (\mathsf{const}\ n)\ cd = \mathsf{LOAD}\ n\ cd \qquad comp\ (\mathsf{add}\ e_0\ e_1)\ cd = comp\ e_1\ (comp\ e_0\ (\mathsf{ADD}\ cd))$$

The machine has then for state a pair cd, S where cd is a code and S is a stack of numbers. The small step semantics for this machine is

$$\frac{}{\mathsf{ADD}\ cd,\ n_1 : n_0 : S \mapsto cd, (n_1 + n_0) : S} \qquad \frac{}{\mathsf{LOAD}\ n\ cd, S \mapsto cd, n : S}$$

We can now state, and prove by induction on e

Theorem 0.1 *For all expression e we have $\forall cd\ S \quad comp\ e\ cd, S \mapsto^* cd, [\![e]\!] : S$*

Krivine Abstract Machine

We define the *terms* (in de Bruijn notation) as

$$t \ ::= \ n \mid \lambda t \mid t\ t$$

namely deBruijn index, or abstraction, or application.

A *value* u is a pair $t\rho$ of a term and an environment, where an *environment* ρ is a list of values.

Krivine Abstract Machine has for states $t \mid \rho \mid S$ where $t\rho$ is a value and S is a stack of values. The small step semantics is

$$\frac{}{0 \mid (t\rho, \nu) \mid S \mapsto t \mid \rho \mid S} \qquad \frac{}{n+1 \mid (u, \nu) \mid S \mapsto n \mid \nu \mid S}$$

$$\frac{}{\lambda t \mid \rho \mid u : S \mapsto t \mid (u, \rho) \mid S}$$

$$\frac{}{t_0\ t_1 \mid \rho \mid S \mapsto t_0 \mid \rho \mid (t_1\rho) : S}$$

So abstraction is "pop" while application is "push".