The Vapor and Vapor-M Operational Semantics

Jens Palsberg

October 10, 2016

1 What are Vapor and Vapor-M?

Vapor and Vapor-M are languages that we will use as intermediate languages when we compile MiniJava to MIPS. The translation steps are: MiniJava → Vapor → Vapor-M → MIPS.

A Vapor program consists of functions that each operates on a heap, global constants, parameters, local variables, and a stack. We will specify Vapor's abstract syntax and operational semantics.

A Vapor-M program consists of functions that each operates on a heap, global constants, global registers, and a stack. We will specify Vapor-M's abstract syntax and operational semantics. Vapor and Vapor-M are closely related. One difference is that in Vapor-M, each function has no parameters, no local variables, and no return value.

2 Notation

2.1 Grammars

The grammars for Vapor and Vapor-M use the following metanotation:

- Nonterminal symbols are words written in *this font*.
- Terminal symbols are written in `this font`, except ⟨STRING⟩, ⟨LABEL⟩, ⟨IDENTIFIER⟩, and ⟨INTEGER_LITERAL⟩.
- A production is of the form *lhs* ::= *rhs*, where *lhs* is a nonterminal symbol and *rhs* is a sequence of nonterminal and terminal symbols, with choices separated by |, and some times using "..." to denote a possibly empty list.
- We will use superscripts and subscripts to distinguish metavariables.

2.2 Rules

We will use the following notation:

$$\frac{hypothesis_1 \qquad hypothesis_2 \quad \ldots \quad hypothesis_n}{conclusion}$$

This is a *rule* that says that if we can derive all of $hypothesis_1$, $hypothesis_2$, ..., $hypothesis_n$, then we can also derive *conclusion*.

A special case arises when $n = 0$: we can write this case as:

$$\frac{}{conclusion}$$

or we can even omit the horizontal bar and write:

$$conclusion$$

We can say that this case is a rule with no hypotheses, or we can call it an *axiom*.

A *derivation* happens when we begin with one or more axioms, then perhaps apply some rules, and finally arrive at a conclusion. Notice that we can organize a derivation as a tree that has the axioms as leaves and the conclusion as the root. We can refer to such as tree as a *derivation tree*.

2.3 Maps

A *map* is a function with finite domain. If M is a map, then $dom(M)$ denotes the domain of M. If $x_1, \ldots, x_r$ are pairwise distinct, then $[x_1 \mapsto y_1, \ldots, x_n \mapsto y_n]$ denotes a map with domain $\{x_1, \ldots, x_n\}$, which maps x_i to y_i, for $i \in 1..n$. If M_1, M_2 are maps, then $M_1 \cdot M_2$ is a map:

$$(M_1 \cdot M_2)(id) = \begin{cases} M_2(id) & \text{if } id \in dom(M_2) \\ M_1(id) & \text{otherwise} \end{cases}$$

Notice that M_2 takes precedence over M_1.

If M is a map and X is a set, then $M \setminus X$ denotes M restricted to $dom(M) \cap X$.

We define a helper function *initmap* that maps a set to a map.

$$(initmap(X))(x) = \begin{cases} 0 & x \in X \\ undefined & \text{otherwise} \end{cases}$$

If M is a map, then we define the notation M^* as follows.

$$M^*(x) \quad = \quad \begin{cases} M(x) & \text{if } x \in dom(M) \\ x & \text{otherwise} \end{cases}$$

2.4 Tuples

$$(\mathit{Tuple}) \quad t \quad ::= \quad \langle y_1, \ldots, y_n \rangle$$

We define a helper function *inittuple* that maps a positive integer to a tuple.

$$inittuple(c) = \langle 0, \ldots, 0 \rangle$$
where the number of 0's in the tuple is c, where $c > 0$

3 Vapor

3.1 Syntax

$$
\begin{array}{rrcl}
(\textit{Program}) & p & ::= & C_1 \ldots C_n \; F_1 \ldots F_m \\
(\textit{ConstDecl}) & C & ::= & \texttt{const} \; l \; l_1 \; \ldots \; l_n \\
(\textit{FunDecl}) & F & ::= & \texttt{func} \; l \; (id_1 \; \ldots \; id_f) \; l_1 \; b_1 \; \ldots \; l_q \; b_q \\
(\textit{Block}) & b & ::= & i_1 \ldots i_n \; j \\
(\textit{Instr}) & i & ::= & id \texttt{ = } o \mid id \texttt{ = } op \texttt{ (} o_1 \; o_2 \texttt{)} \mid id \texttt{ = } m \mid m \texttt{ = } id \\
 & & \mid & \texttt{if0} \; o \; \texttt{goto} \; l \mid id \texttt{ = call } o \texttt{ (} o_1 \; \ldots \; o_f \texttt{)} \\
 & & \mid & id \texttt{ = HeapAllocZ (} o \texttt{)} \mid \texttt{PrintIntS (} o \texttt{)} \mid \texttt{Error (} s \texttt{)} \\
(\textit{Jump}) & j & ::= & \texttt{goto} \; l \mid \texttt{ret} \; o \mid \texttt{ret} \\
(\textit{MemRef}) & m & ::= & \texttt{[} id \texttt{ + c]} \\
(\textit{Operator}) & op & ::= & \texttt{Add} \mid \texttt{Sub} \mid \texttt{MulS} \mid \texttt{Eq} \mid \texttt{LtS} \\
(\textit{Operand}) & o & ::= & l \mid c \mid id \\
(\textit{StringLiteral}) & s & ::= & \langle \mathsf{STRING} \rangle \\
(\textit{Label}) & l & ::= & \langle \mathsf{LABEL} \rangle \\
(\textit{IntegerLiteral}) & c & ::= & \langle \mathsf{INTEGER\_LITERAL} \rangle \\
(\textit{Identifier}) & id & ::= & \langle \mathsf{IDENTIFIER} \rangle
\end{array}
$$

3.2 Helper Functions

Defined Variables. We define a helper function $defined$ that maps a block to the set of local variables that the blocks assigns. We will overload $defined$ and define it also for instructions.

$$defined(\; i_1 \ldots \; i_n \; j \;) \;=\; (defined(i_1) \cup \ldots \; defined(i_n))$$

$$
\begin{array}{rcl}
defined(\; id \texttt{ = } o \;) & = & \{id\} \\
defined(\; id \texttt{ = } op \texttt{ (} o_1 \; o_2 \texttt{) }) & = & \{id\} \\
defined(\; id \texttt{ = } m \;) & = & \{id\} \\
defined(\; m \texttt{ = } id \;) & = & \emptyset \\
defined(\; \texttt{if0} \; o \; \texttt{goto} \; l \;) & = & \emptyset \\
defined(\; id \texttt{ = call } o \texttt{ (} o_1 \; \ldots \; o_f \texttt{) }) & = & \{id\} \\
defined(\; \texttt{HeapAllocZ (} o \texttt{) }) & = & \emptyset \\
defined(\; \texttt{PrintIntS (} o \texttt{) }) & = & \emptyset \\
defined(\; \texttt{Error (} s \texttt{) }) & = & \emptyset
\end{array}
$$

Initialization of Constants. We define a helper function $initconst$ that maps a constant declaration to a map.

$$initconst(\texttt{const} \; l \; l_1 \; \ldots \; l_n \;) \;=\; [l \mapsto \langle l_1 \ldots \; l_n \rangle]$$

Initialization of Functions. We define a helper function $initfun$ that maps a function declaration to a map.

$$
\begin{array}{l}
initfun(F) \;=\; [l \mapsto F, \; l_1 \mapsto b_1, \; \ldots, \; l_q \mapsto b_q] \\
\quad \text{where} \\
\qquad F \;=\; \texttt{func} \; l \; (id_1 \; \ldots \; id_f) \; l_1 \; b_1 \; \ldots \; l_q \; b_q
\end{array}
$$

3.3 Program States

Vapor has three kinds of values: labels l, heap addresses (l, c), and integers c. We use v to range over values.

A program state (G, H, E, b) has four components. Intuitively, G is a *global table* that represents the constants, functions, and blocks; H is the *heap*; E is an *environment* that represents the parameters and local variables; and b is the block that is executing right now.

The Global Table. The global table is a map from labels to either tuples, functions, or blocks.

$$\begin{array}{rrcl} (\mathit{GlobalData}) & d & ::= & t \mid F \mid b \\ (\mathit{GlobalTable}) & G & ::= & [l_1 \mapsto d_1, \ldots, l_n \mapsto d_n] \end{array}$$

The Heap. A heap is a map from labels to tuples. We use H to range over heaps. A *heap address* is a pair of the form (l, c), where l is a label and c is an integer such that $c \geq 0$ and c is divisible by 4.

The Environment. An environment represents the parameters, and local variables. An environment is a map from identifiers to values. We use E to range over environments.

The Initial Program State. Consider a program

$$C_1 \ldots\ C_n\ F_1 \ldots\ F_m$$

where

$$F_1 \quad = \quad \texttt{func}\ id\ \texttt{(}\ \texttt{)}\ l_1\ b_1\ \ \texttt{...}\ \ l_q\ b_q$$

Notice that F_1 has no parameters. The initial program state is (G, H, E, b), where:

$$\begin{array}{rcl} G & = & initconst(C_1) \cdot \ldots\ \cdot initconst(C_n) \cdot initfun(F_1) \cdot \ldots\ \cdot\ initfun(F_m) \\ H & = & [\,] \\ E & = & initmap(\ (defined(b_1) \cup \ldots\ \cup defined(b_q))\ ) \\ b & = & b_1 \end{array}$$

3.4 Semantics

Single Steps.

$$(G, H, E, id\ \texttt{=}\ o \quad b') \longmapsto (G, H, E \cdot [id \mapsto E^*(o)], b') \tag{1}$$

$$\begin{array}{l}(G, H, E, id\ \texttt{=}\ \texttt{Add}\ \texttt{(}\ o_1\ \ o_2\ \texttt{)} \quad b') \longmapsto (G, H, E \cdot [id \mapsto (c_1 + c_2)], b') \\ \qquad \text{if } E^*(o_1) = c_1 \text{ and } E^*(o_2) = c_2\end{array} \tag{2}$$

$$\begin{array}{l}(G, H, E, id\ \texttt{=}\ \texttt{Add}\ \texttt{(}\ o_1\ \ o_2\ \texttt{)} \quad b') \longmapsto (G, H, E \cdot [id \mapsto (l, c_1 + c_2)], b') \\ \qquad \text{if } E^*(o_1) = (l, c_1) \text{ and } E^*(o_2) = c_2 \\ \qquad \text{where } c_2 \geq 0 \text{ and } c_2 \text{ is divisible by } 4\end{array} \tag{3}$$

$$\begin{array}{l}(G, H, E, id\ \texttt{=}\ \texttt{Sub}\ \texttt{(}\ o_1\ \ o_2\ \texttt{)} \quad b') \longmapsto (G, H, E \cdot [id \mapsto (c_1 - c_2)], b') \\ \qquad \text{if } E^*(o_1) = c_1 \text{ and } E^*(o_2) = c_2\end{array} \tag{4}$$

$$\begin{array}{l}(G, H, E, id\ \texttt{=}\ \texttt{MulS}\ \texttt{(}\ o_1\ \ o_2\ \texttt{)} \quad b') \longmapsto (G, H, E \cdot [id \mapsto (c_1 \times c_2)], b') \\ \qquad \text{if } E^*(o_1) = c_1 \text{ and } E^*(o_2) = c_2\end{array} \tag{5}$$

$$\begin{array}{l}(G, H, E, id\ \texttt{=}\ \texttt{Eq}\ \texttt{(}\ o_1\ \ o_2\ \texttt{)} \quad b') \longmapsto (G, H, E \cdot [id \mapsto 1], b') \\ \qquad \text{if } E^*(o_1) = c_1 \text{ and } E^*(o_2) = c_2 \text{ and } c_1 = c_2\end{array} \tag{6}$$

$$\begin{array}{l}(G, H, E, id\ \texttt{=}\ \texttt{Eq}\ \texttt{(}\ o_1\ \ o_2\ \texttt{)} \quad b') \longmapsto (G, H, E \cdot [id \mapsto 0], b') \\ \qquad \text{if } E^*(o_1) = c_1 \text{ and } E^*(o_2) = c_2 \text{ and } c_1 \neq c_2\end{array} \tag{7}$$

$$\begin{array}{l}(G, H, E, id\ \texttt{=}\ \texttt{LtS}\ \texttt{(}\ o_1\ \ o_2\ \texttt{)} \quad b') \longmapsto (G, H, E \cdot [id \mapsto 1], b') \\ \qquad \text{if } E^*(o_1) = c_1 \text{ and } E^*(o_2) = c_2 \text{ and } c_1 < c_2\end{array} \tag{8}$$

$$\begin{array}{l}(G, H, E, id\ \texttt{=}\ \texttt{LtS}\ \texttt{(}\ o_1\ \ o_2\ \texttt{)} \quad b') \longmapsto (G, H, E \cdot [id \mapsto 0], b') \\ \qquad \text{if } E^*(o_1) = c_1 \text{ and } E^*(o_2) = c_2 \text{ and } c_1 \geq c_2\end{array} \tag{9}$$

$$\begin{array}{l}(G, H, E, id\ \texttt{=}\ \texttt{[}\ id'\ \texttt{+}\ \texttt{c}\ \texttt{]} \quad b') \longmapsto (G, H, E \cdot [id \mapsto v_{c'+c}], b') \\ \qquad \text{where } E^*(id') = (l, c') \\ \qquad \text{where } c \geq 0 \text{ and } c \text{ is divisible by } 4 \\ \qquad \text{where } H(l) = \langle v_0, v_4, \ldots, v_{c'+c}, \ldots\ \ v_n \rangle \text{ or else } G(l) = \langle v_0, v_4, \ldots, v_{c'+c}, \ldots\ \ v_n \rangle \\ \qquad \text{where } c' + c \leq n\end{array} \tag{10}$$

$$\begin{array}{l}(G, H, E, \texttt{[}\ id\ \texttt{+}\ \texttt{c}\ \texttt{]}\ \texttt{=}\ id' \quad b') \longmapsto (G, H[l \mapsto t'], E, b') \\ \qquad \text{where } E^*(id) = (l, c') \\ \qquad \text{where } c \geq 0 \text{ and } c \text{ is divisible by } 4 \\ \qquad \text{where } H(l) = \langle v_0, v_4, \ldots\ \ v_n \rangle \\ \qquad \text{where } t' = \langle v_0, v_4, \ldots\ \ v_{c-4}\ E^*(id')\ v_{c+4} \ldots\ \ v_n \rangle \\ \qquad \text{where } c' + c \leq n\end{array} \tag{11}$$

$$(G, H, E, \texttt{if0}\ o\ \texttt{goto}\ l \quad b') \longmapsto (G, H, E, b'')$$
$$\text{if } E^*(o) = 0 \text{ and } G(l) = b'', \tag{12}$$
$$\text{and } l\ b'' \text{ and } \texttt{if0}\ o\ \texttt{goto}\ l \text{ are in the body of the same function}$$

$$(G, H, E, \texttt{if0}\ o\ \texttt{goto}\ l \quad b') \longmapsto (G, H, E, b')$$
$$\text{if } E^*(o) = c \text{ and } c \neq 0 \tag{13}$$

$$\frac{(G, H, E^{init}, b_1) \longmapsto (G, H', E', \texttt{ret}\ o')}{(G, H, E, id\ \texttt{=}\ \texttt{call}\ o\ \texttt{(}\ o_1\ \texttt{...}\ o_f\ \texttt{)} \quad b') \longmapsto (G, H', E \cdot [id \mapsto E'^*(o')], b')}$$
$$\text{where } E^*(o) = l$$
$$\text{where } G(l) = \texttt{func}\ l\ \texttt{(}id_1\ \texttt{...}\ id_f\texttt{)}\ l_1\ b_1\ \texttt{...}\ l_q\ b_q \tag{14}$$
$$\text{where } E^{init} = [id_1 \mapsto E^*(o_1), \ldots, id_f \mapsto E^*(o_f)] \cdot initmap(a)$$
$$\text{where } a = (defined(b_1) \cup \ldots \cup defined(b_q)) \setminus \{id_1, \ldots, id_f\}$$

$$(G, H, E, id\ \texttt{=}\ \texttt{HeapAllocZ}\ \texttt{(}\ o\ \texttt{)} \quad b') \longmapsto (G, H \cup [l \mapsto t], E \cdot [id \mapsto (l, 0)], E, b')$$
$$\text{where } l \notin dom(G, H) \text{ and } E^*(o) \text{ is a positive integer that is divisible by } 4 \tag{15}$$
$$\text{where } t = inittuple(\tfrac{E^*(o)}{4})$$

$$(G, H, E, \texttt{PrintIntS}\ \texttt{(}\ o\ \texttt{)} \quad b') \longmapsto (G, H, E, b')$$
$$\text{where } E^*(o) = c \tag{16}$$
$$\text{and display } c \text{ on the screen}$$

$$(G, H, E, \texttt{Error}\ \texttt{(}\ s\ \texttt{)} \quad b')$$
$$\text{display } s \text{ on the screen and stop execution} \tag{17}$$

$$(G, H, E, \texttt{goto}\ l) \longmapsto (G, H, E, b')$$
$$\text{if } G(l) = b', \tag{18}$$
$$\text{and } l\ b' \text{ and } \texttt{goto}\ l \text{ are in the body of the same function}$$

Multiple Steps.

$$\frac{(G, H, E, b) \longmapsto (G', H', E', b') \qquad (G', H', E', b') \longmapsto (G'', H'', E'', b'')}{(G, H, E, b) \longmapsto (G'', H'', E'', b'')} \tag{19}$$

4 Vapor-M

4.1 Syntax

$$
\begin{array}{rrcl}
(\textit{Program}) & p & ::= & C_1 \ldots\ C_n\ F_1 \ldots\ F_m \\
(\textit{ConstDecl}) & C & ::= & \texttt{const}\ l\ l_1\ \ldots\ l_n \\
(\textit{FunDecl}) & F & ::= & \texttt{func}\ l\ \texttt{[in}\ c_1\texttt{, out}\ c_2\texttt{, local}\ c_3\texttt{]}\ l_1\ b_1\ \ldots\ l_q\ b_q \\
(\textit{Block}) & b & ::= & i_1 \ldots\ i_n\ j \\
(\textit{Instr}) & i & ::= & id\ \texttt{=}\ o \ \mid\ id\ \texttt{=}\ op\ \texttt{(}\ o_1\ o_2\ \texttt{)} \ \mid\ id\ \texttt{=}\ m \ \mid\ m\ \texttt{=}\ id \\
 & & \mid & \texttt{if0}\ o\ \texttt{goto}\ l \ \mid\ \texttt{call}\ o \\
 & & \mid & id\ \texttt{=}\ \texttt{HeapAllocZ}\ \texttt{(}\ o\ \texttt{)} \ \mid\ \texttt{PrintIntS}\ \texttt{(}\ o\ \texttt{)} \ \mid\ \texttt{Error}\ \texttt{(}\ s\ \texttt{)} \\
(\textit{Jump}) & j & ::= & \texttt{goto}\ l \ \mid\ \texttt{ret} \\
(\textit{MemRef}) & m & ::= & \texttt{[}\ id\ \texttt{+ c]} \ \mid\ \texttt{in [}\ c\ \texttt{]} \mid \texttt{out [}\ c\ \texttt{]} \mid \texttt{local [}\ c\ \texttt{]} \\
(\textit{Operator}) & op & ::= & \texttt{Add} \mid \texttt{Sub} \mid \texttt{MulS} \mid \texttt{Eq} \mid \texttt{LtS} \\
(\textit{Operand}) & o & ::= & l \mid c \mid id \\
(\textit{StringLiteral}) & s & ::= & \langle \textsf{STRING} \rangle \\
(\textit{Label}) & l & ::= & \langle \textsf{LABEL} \rangle \\
(\textit{IntegerLiteral}) & c & ::= & \langle \textsf{INTEGER\_LITERAL} \rangle \\
(\textit{Identifier}) & id & ::= & \langle \textsf{IDENTIFIER} \rangle
\end{array}
$$

4.2 Helper Functions

Defined Variables. We define a helper function $defined$ that maps a block to the set of local variables that the blocks assigns. We will overload $defined$ and define it also for instructions.

$$defined(\ i_1 \ldots\ i_n\ j\ ) \quad = \quad (defined(i_1) \cup \ldots\ defined(i_n))$$

$$
\begin{array}{rcl}
defined(\ id\ \texttt{=}\ o\ ) & = & \{id\} \\
defined(\ id\ \texttt{=}\ op\ \texttt{(}\ o_1\ o_2\ \texttt{)}\ ) & = & \{id\} \\
defined(\ id\ \texttt{=}\ m\ ) & = & \{id\} \\
defined(\ m\ \texttt{=}\ id\ ) & = & \emptyset \\
defined(\ \texttt{if0}\ o\ \texttt{goto}\ l\ ) & = & \emptyset \\
defined(\ \texttt{call}\ o\ ) & = & \emptyset \\
defined(\ \texttt{HeapAllocZ}\ \texttt{(}\ o\ \texttt{)}\ ) & = & \emptyset \\
defined(\ \texttt{PrintIntS}\ \texttt{(}\ o\ \texttt{)}\ ) & = & \emptyset \\
defined(\ \texttt{Error}\ \texttt{(}\ s\ \texttt{)}\ ) & = & \emptyset
\end{array}
$$

Initialization of Constants. We define a helper function $initconst$ that maps a constant declaration to a map.

$$initconst(\texttt{const}\ l\ l_1\ \ldots\ l_n\ ) \quad = \quad [l \mapsto \langle l_1 \ldots\ l_n \rangle]$$

Initialization of Functions. We define a helper function $initfun$ that maps a function declaration to a map.

$$
\begin{array}{rcl}
initfun(F) & = & [l \mapsto F,\ l_1 \mapsto b_1,\ \ldots,\ l_q \mapsto b_q] \\
\text{where} & & \\
F & = & \texttt{func}\ l\ \texttt{[in}\ c_1\texttt{, out}\ c_2\texttt{, local}\ c_3\texttt{]}\ l_1\ b_1\ \ldots\ l_q\ b_q
\end{array}
$$

4.3 Program States

Vapor-M has three kinds of values: labels l, heap addresses (l, c), and integers c. We use v to range over values.

A program state (G, H, R, S, b) has five components. Intuitively, G is a *global table* that represents the constants, functions, and blocks; H is the *heap*; R is the *register file*; S is the *stack*; and b is the block that is executing right now.

The Global Table. The global table is a map from labels to either tuples, functions, or blocks.

$$\begin{array}{rrcl} (\mathit{GlobalData}) & d & ::= & t \mid F \mid b \\ (\mathit{GlobalTable}) & G & ::= & [l_1 \mapsto d_1, \ldots, l_n \mapsto d_n] \end{array}$$

The Heap. A heap is a map from labels to tuples. We use H to range over heaps. A *heap address* is a pair of the form (l, c), where l is a label and c is an integer such that $c \geq 0$ and c is divisible by 4.

The Register File. The set *registers* consists of 23 identifiers that are akin to the names of 23 MIPS registers.

$$\mathit{registers} \quad = \quad \{\mathtt{s0}, \ldots, \mathtt{s7}, \mathtt{t0}, \ldots, \mathtt{t8}, \mathtt{a0}, \ldots, \mathtt{a3}, \mathtt{v0}, \mathtt{v1}\}$$

Intuitively, the elements of *registers* are the global registers. A register file is a map from *registers* to values. We use R to range over register files.

The Stack.

$$(\mathit{Stack}) \quad S \quad ::= \quad \mathit{empty} \mid S \odot t$$

The Initial Program State. Consider a program

$$C_1 \ldots\ C_n\ F_1 \ldots\ F_m$$

where

$$F_1 \quad = \quad \mathtt{func}\ \mathit{id}\ \mathtt{[in\ 0,\ out}\ c_2\mathtt{,\ local}\ c_3\mathtt{]}\ l_1\ b_1\ \mathtt{...}\ l_q\ b_q$$

Notice that F_1 has no parameters and has "`[in 0]`". The initial program state is (G, H, R, S, b), where:

$$\begin{array}{rcl} G & = & \mathit{initconst}(C_1) \cdot \ldots \cdot \mathit{initconst}(C_n) \cdot \mathit{initfun}(F_1) \cdot \ldots \cdot \mathit{initfun}(F_m) \\ H & = & [\,] \\ R & = & \mathit{initmap}(\mathit{registers}) \\ S & = & \mathit{empty} \odot \mathit{inittuple}(c_3) \odot \mathit{inittuple}(c_2) \\ b & = & b_1 \end{array}$$

4.4 Semantics

Single Steps.

$$(G, H, R, S, id \texttt{ = } o \quad b') \longmapsto (G, H, R \cdot [id \mapsto R^*(o)], S, b') \tag{20}$$

$$\begin{aligned}&(G, H, R, S, id \texttt{ = Add (} o_1 \;\; o_2 \texttt{)} \quad b') \longmapsto (G, H, R \cdot [id \mapsto (c_1 + c_2), S, b')\\ &\qquad \text{if } R^*(o_1) = c_1 \text{ and } R^*(o_2) = c_2\end{aligned} \tag{21}$$

$$\begin{aligned}&(G, H, R, S, id \texttt{ = Add (} o_1 \;\; o_2 \texttt{)} \quad b') \longmapsto (G, H, R \cdot [id \mapsto (l, c_1 + c_2)], S, b')\\ &\qquad \text{if } R^*(o_1) = (l, c_1) \text{ and } R^*(o_2) = c_2\\ &\qquad \text{where } c_2 \geq 0 \text{ and } c_2 \text{ is divisible by 4}\end{aligned} \tag{22}$$

$$\begin{aligned}&(G, H, R, S, id \texttt{ = Sub (} o_1 \;\; o_2 \texttt{)} \quad b') \longmapsto (G, H, R \cdot [id \mapsto (c_1 - c_2), S, b')\\ &\qquad \text{if } R^*(o_1) = c_1 \text{ and } R^*(o_2) = c_2\end{aligned} \tag{23}$$

$$\begin{aligned}&(G, H, R, S, id \texttt{ = MulS (} o_1 \;\; o_2 \texttt{)} \quad b') \longmapsto (G, H, R \cdot [id \mapsto (c_1 \times c_2), S, b')\\ &\qquad \text{if } R^*(o_1) = c_1 \text{ and } R^*(o_2) = c_2\end{aligned} \tag{24}$$

$$\begin{aligned}&(G, H, R, S, id \texttt{ = Eq (} o_1 \;\; o_2 \texttt{)} \quad b') \longmapsto (G, H, R \cdot [id \mapsto 1], S, b')\\ &\qquad \text{if } R^*(o_1) = c_1 \text{ and } R^*(o_2) = c_2 \text{ and } c_1 = c_2\end{aligned} \tag{25}$$

$$\begin{aligned}&(G, H, R, S, id \texttt{ = Eq (} o_1 \;\; o_2 \texttt{)} \quad b') \longmapsto (G, H, R \cdot [id \mapsto 0], S, b')\\ &\qquad \text{if } R^*(o_1) = c_1 \text{ and } R^*(o_2) = c_2 \text{ and } c_1 \neq c_2\end{aligned} \tag{26}$$

$$\begin{aligned}&(G, H, R, S, id \texttt{ = LtS (} o_1 \;\; o_2 \texttt{)} \quad b') \longmapsto (G, H, R \cdot [id \mapsto 1], S, b')\\ &\qquad \text{if } R^*(o_1) = c_1 \text{ and } R^*(o_2) = c_2 \text{ and } c_1 < c_2\end{aligned} \tag{27}$$

$$\begin{aligned}&(G, H, R, S, id \texttt{ = LtS (} o_1 \;\; o_2 \texttt{)} \quad b') \longmapsto (G, H, R \cdot [id \mapsto 0], S, b')\\ &\qquad \text{if } R^*(o_1) = c_1 \text{ and } R^*(o_2) = c_2 \text{ and } c_1 \geq c_2\end{aligned} \tag{28}$$

$$\begin{aligned}&(G, H, R, S, id \texttt{ = [} id' \texttt{ + c]} \quad b') \longmapsto (G, H, R \cdot [id \mapsto v_{c'+c}], S, b')\\ &\qquad \text{where } R * (id') = (l, c')\\ &\qquad \text{where } c \geq 0 \text{ and } c \text{ is divisible by 4}\\ &\qquad \text{where } H(l) = \langle v_0, v_4, \ldots, v_{c'+c}, \ldots \;\; v_n\rangle \text{ or else } G(l) = \langle v_0, v_4, \ldots, v_{c'+c}, \ldots \;\; v_n\rangle\\ &\qquad \text{where } c' + c \leq n\end{aligned} \tag{29}$$

$$\begin{aligned}&(G, H, R, S, id \texttt{ = in [} c \texttt{]} \quad b') \longmapsto (G, H, R \cdot [id \mapsto v_c], S, b')\\ &\qquad \text{where } S = S' \odot t_{in} \odot t_{local} \odot t_{out}\\ &\qquad \text{where } t_{in} = \langle v_m \ldots \;\; v_c \ldots \;\; v_1, v_0\rangle\end{aligned} \tag{30}$$

$$\begin{aligned}&(G, H, R, S, id \texttt{ = out [} c \texttt{]} \quad b') \longmapsto (G, H, R \cdot [id \mapsto v_c], S, b')\\ &\qquad \text{where } S = S' \odot t_{in} \odot t_{local} \odot t_{out}\\ &\qquad \text{where } t_{out} = \langle v_m \ldots \;\; v_c \ldots \;\; v_1, v_0\rangle\end{aligned} \tag{31}$$

$$\begin{aligned}&(G, H, R, S, id \texttt{ = local [} c \texttt{]} \quad b') \longmapsto (G, H, R \cdot [id \mapsto v_c], S, b')\\ &\qquad \text{where } S = S' \odot t_{in} \odot t_{local} \odot t_{out}\\ &\qquad \text{where } t_{local} = \langle v_m \ldots \;\; v_c \ldots \;\; v_1, v_0\rangle\end{aligned} \tag{32}$$

$$\begin{aligned}&(G, H, R, S, \texttt{[} id \texttt{ + c] = } id' \quad b') \longmapsto (G, H[l \mapsto t'], R, S, b')\\ &\qquad \text{where } R^*(id) = (l, c')\\ &\qquad \text{where } c \geq 0 \text{ and } c \text{ is divisible by 4}\\ &\qquad \text{where } H(l) = \langle v_0, v_4, \ldots \;\; v_n\rangle\\ &\qquad \text{where } t' = \langle v_0, v_4, \ldots \;\; v_{c-4} \;\; R^*(id') \;\; v_{c+4} \ldots \;\; v_n\rangle\\ &\qquad \text{where } c' + c \leq n\end{aligned} \tag{33}$$

$$(G, H, R, S, \texttt{in [} c \texttt{] = } id \quad b') \longmapsto (G, H, R, S \odot t'_{in} \odot t_{local} \odot t_{out}, b') \tag{34}$$
where $S = S' \odot t_{in} \odot t_{local} \odot t_{out}$
where $t_{in} = \langle v_m \ldots\ v_c \ldots\ v_1, v_0 \rangle$
where $t'_{in} = \langle v_m \ldots\ R^*(id) \ldots\ v_1, v_0 \rangle$

$$(G, H, R, S, \texttt{out [} c \texttt{] = } id \quad b') \longmapsto (G, H, R, S \odot t_{in} \odot t_{local} \odot t'_{out}, b') \tag{35}$$
where $S = S' \odot t_{in} \odot t_{local} \odot t_{out}$
where $t_{out} = \langle v_m \ldots\ v_c \ldots\ v_1, v_0 \rangle$
where $t'_{out} = \langle v_m \ldots\ R^*(id) \ldots\ v_1, v_0 \rangle$

$$(G, H, R, S, \texttt{local [} c \texttt{] = } id \quad b') \longmapsto (G, H, R, S \odot t_{in} \odot t'_{local} \odot t_{out}, b') \tag{36}$$
where $S = S' \odot t_{in} \odot t_{local} \odot t_{out}$
where $t_{local} = \langle v_m \ldots\ v_c \ldots\ v_1, v_0 \rangle$
where $t'_{local} = \langle v_m \ldots\ R^*(id) \ldots\ v_1, v_0 \rangle$

$$(G, H, R, S, \texttt{if0 } o \texttt{ goto } l \quad b') \longmapsto (G, H, R, S, b'') \tag{37}$$
if $R^*(o) = 0$ and $G(l) = b''$,
and $l\ b''$ and $\texttt{if0 } o \texttt{ goto } l$ are in the body of the same function

$$(G, H, R, S, \texttt{if0 } o \texttt{ goto } l \quad b') \longmapsto (G, H, R, S, b') \tag{38}$$
if $R^*(o) = c$ and $c \neq 0$

$$\frac{(G, H, R, S \odot \mathit{inittuple}(c_3) \odot \mathit{inittuple}(c_2), b_1) \longmapsto (G, H', R', S' \odot t_3 \odot t_2, \texttt{ret})}{(G, H, R, S, \texttt{call } o \quad b') \longmapsto (G, H', R', S', b')} \tag{39}$$
where $R^*(o) = l$
where $G(l) = \texttt{func } l \texttt{ [in } c_1\texttt{, out } c_2\texttt{, local } c_3\texttt{] } l_1\ b_1\ \ldots\ l_q\ b_q$

$$(G, H, R, S, id \texttt{ = HeapAllocZ (} o \texttt{)} \quad b') \longmapsto (G, H \cup [l \mapsto t], R \cdot [id \mapsto (l, 0)], S, b') \tag{40}$$
where $l \notin dom(G, H)$ and $R^*(o)$ is a positive integer that is divisible by 4
where $t = \mathit{inittuple}(\frac{R^*(o)}{4})$

$$(G, H, R, S, \texttt{PrintIntS (} o \texttt{)} \quad b') \longmapsto (G, H, R, S, b') \tag{41}$$
where $R^*(o) = c$
and display c on the screen

$$(G, H, R, S, \texttt{Error (} s \texttt{)} \quad b') \tag{42}$$
display s on the screen and stop execution

$$(G, H, R, S, \texttt{goto } l) \longmapsto (G, H, R, S, b') \tag{43}$$
if $G(l) = b'$,
and $l\ b'$ and $\texttt{goto } l$ and in the body of the same function

Multiple Steps.

$$\frac{(G, H, R, S, b) \longmapsto (G', H', R', S', b') \qquad (G', H', R', S', b') \longmapsto (G'', H'', R'', S'', b'')}{(G, H, R, S, b) \longmapsto (G'', H'', R'', S'', b'')} \tag{44}$$