Sparrow and Sparrow-V

Jens Palsberg

September 12, 2020

1 What are Sparrow and Sparrow-V?

Sparrow and Sparrow-V are closely related languages that we will use as intermediate languages when we compile MiniJava to RISC-V. The translation steps are: MiniJava → Sparrow → Sparrow-V → RISC-V.

We will specify Sparrow and Sparrow-V's abstract syntax and operational semantics. Briefly, a Sparrow program consists of functions that each operates on a heap, parameters, and local variables. A Sparrow-V program consists of functions that each operates on a heap, global registers, parameters, and local variables.

2 Notation

Grammars. The grammars for Sparrow and Sparrow-V use the following metanotation:

- Nonterminal symbols are words written in *this font*.
- Terminal symbols are written in `this font`, except ⟨FUNCTIONNAME⟩, ⟨LABEL⟩, ⟨IDENTIFIER⟩, ⟨INTEGER_LITERAL⟩, and ⟨STRING⟩.
- 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.

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 $n = 0$, that is, a rule with no hypotheses, also known as an *axiom*; we may omit the horizontal bar and write: *conclusion*.

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 refer to such as tree as a *derivation tree*.

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.

A *tuple* t is a map with a fixed domain of k integers $0, 4, 8, \ldots, 4(k-1)$.

3 Sparrow

3.1 Syntax

$$
\begin{array}{rrcl}
(\textit{Program}) & p & ::= & F_1 \ \ldots \ F_m \\
(\textit{FunDecl}) & F & ::= & \texttt{func}\ f\ \texttt{(}id_1\ \texttt{...}\ id_f\texttt{)}\ b \\
(\textit{Block}) & b & ::= & i_1\ \ldots\ i_n\ \texttt{return}\ id \\
(\textit{Instruction}) & i & ::= & l: \mid id\ \texttt{=}\ c \mid id\ \texttt{=}\ \texttt{@}\ f \\
& & \mid & id\ \texttt{=}\ id\ \texttt{+}\ id \mid id\ \texttt{=}\ id\ \texttt{-}\ id \mid id\ \texttt{=}\ id\ \texttt{*}\ id \mid id\ \texttt{=}\ id\ \texttt{<}\ id \\
& & \mid & id\ \texttt{=}\ \texttt{[}\ id\ \texttt{+}\ c\ \texttt{]} \mid \texttt{[}\ id\ \texttt{+}\ c\ \texttt{]}\ \texttt{=}\ id \mid id\ \texttt{=}\ id \\
& & \mid & id\ \texttt{=}\ \texttt{alloc (}id\texttt{)} \mid \texttt{print (}id\texttt{)} \mid \texttt{error (}s\texttt{)} \mid \texttt{goto}\ l \\
& & \mid & \texttt{if0}\ id\ \texttt{goto}\ l \mid id\ \texttt{=}\ \texttt{call}\ id\ \texttt{(}id\ \texttt{...}\ id\texttt{)} \\
(\textit{FunctionName}) & f & ::= & \langle\mathsf{FUNCTIONNAME}\rangle \\
(\textit{Label}) & l & ::= & \langle\mathsf{LABEL}\rangle \\
(\textit{Identifier}) & id & ::= & \langle\mathsf{IDENTIFIER}\rangle \setminus \{\texttt{a2},\ldots,\texttt{a7},\texttt{s1},\ldots,\texttt{s11},\texttt{t0},\ldots,\texttt{t5}\} \\
(\textit{IntegerLiteral}) & c & ::= & \langle\mathsf{INTEGER\_LITERAL}\rangle \\
(\textit{StringLiteral}) & s & ::= & \langle\mathsf{STRING}\rangle
\end{array}
$$

We require that F_1 has no parameters. Also, in the instructions id `= [` id `+` c `]` and `[` id `+` c `] =` id, we require that $c \geq 0$ and that c is divisible by 4.

3.2 Helper Function

We define $\mathsf{find}(b, l) = b'$ such that $b = i_1 \ldots i_q \;\; l: \;\; b'$, that is, b' is the suffix of b that follows label l.

3.3 Values, Heaps, Environments, and Program States

Values. Sparrow has three kinds of values: integers c, heap addresses with offsets (a, c), and function names f. We use v to range over values.

The Heap. A heap H is a map from heap addresses a to tuples of values. A *heap address with offset* is a pair of the form (a, c), where a is a heap address and c is an integer such that $c \geq 0$ and c is divisible by 4.

The Environment. An environment E represents the parameters and local variables. An environment is a map from identifiers to values.

Program States. A program state $(p, H, b^\bullet, E, b)$ has five components. Intuitively, p is the entire program, H is the heap, $b^\bullet$ is the entire body of the function that is executing right now, E is an environment that represents the parameters and local variables, and b is the block that is executing right now.

The Initial Program State. Consider a program $p = F_1 \ \ldots \ F_m$ where $F_1 = \texttt{func}\ id\ \texttt{( )}\ b$. The initial program state is $(p, [\,], b, [\,], b)$.

3.4 Semantics

The execution relation is the reflexive, transitive closure of $\longmapsto$, which we define as follows.

$$
\begin{array}{rcl}
(p, H, b^\bullet, E,\ l :\ \ b) & \longmapsto & (p, H, b^\bullet, E, b) \\
(p, H, b^\bullet, E,\ id\ \texttt{=}\ c\ \ b) & \longmapsto & (p, H, b^\bullet, E \cdot [id \mapsto c], b) \\
(p, H, b^\bullet, E,\ id\ \texttt{=}\ \texttt{@}\ f\ \ b) & \longmapsto & (p, H, b^\bullet, E \cdot [id \mapsto f], b) \\
(p, H, b^\bullet, E,\ id\ \texttt{=}\ id_1\ \texttt{+}\ id_2\ \ b) & \longmapsto & (p, H, b^\bullet, E \cdot [id \mapsto (c_1 + c_2)], b) \\
 & & \text{if } E(id_1) = c_1 \wedge E(id_2) = c_2 \\
(p, H, b^\bullet, E,\ id\ \texttt{=}\ id_1\ \texttt{+}\ id_2\ \ b) & \longmapsto & (p, H, b^\bullet, E \cdot [id \mapsto (a_1, c_1 + c_2)], b) \\
 & & \text{if } E(id_1) = (a_1, c_1) \wedge E(id_2) = c_2 \\
 & & \text{and } c_2 \geq 0 \wedge c_2 \text{ is divisible by } 4 \\
(p, H, b^\bullet, E,\ id\ \texttt{=}\ id_1\ \texttt{-}\ id_2\ \ b) & \longmapsto & (p, H, b^\bullet, E \cdot [id \mapsto (c_1 - c_2)], b) \\
 & & \text{if } E(id_1) = c_1 \wedge E(id_2) = c_2 \\
(p, H, b^\bullet, E,\ id\ \texttt{=}\ id_1\ \texttt{*}\ id_2\ \ b) & \longmapsto & (p, H, b^\bullet, E \cdot [id \mapsto (c_1 \times c_2), b) \\
 & & \text{if } E(id_1) = c_1 \wedge E(id_2) = c_2 \\
(p, H, b^\bullet, E,\ id\ \texttt{=}\ id_1\ \texttt{<}\ id_2\ \ b) & \longmapsto & (p, H, b^\bullet, E \cdot [id \mapsto 1], b) \\
 & & \text{if } E(id_1) = c_1 \wedge E(id_2) = c_2 \wedge c_1 < c_2 \\
(p, H, b^\bullet, E,\ id\ \texttt{=}\ id_1\ \texttt{<}\ id_2\ \ b) & \longmapsto & (p, H, b^\bullet, E \cdot [id \mapsto 0], b) \\
 & & \text{if } E(id_1) = c_1 \wedge E(id_2) = c_2 \wedge c_1 \geq c_2 \\
(p, H, b^\bullet, E,\ id\ \texttt{=}\ \texttt{[}\ id_1\ \texttt{+}\ c\ \texttt{]}\ \ b) & \longmapsto & (p, H, b^\bullet, E \cdot [id \mapsto (H(a_1))(c_1 + c)], b) \\
 & & \text{if } E(id_1) = (a_1, c_1) \wedge (c_1 + c) \in dom(H(a_1)) \\
(p, H, b^\bullet, E,\ \texttt{[}\ id_1\ \texttt{+}\ c\ \texttt{]}\ \texttt{=}\ id\ \ b) & \longmapsto & (p, H \cdot [a_1 \mapsto t], b^\bullet, E, b) \\
 & & \text{if } E(id_1) = (a_1, c_1) \wedge (c_1 + c) \in dom(H(a_1)) \\
 & & \text{and } t = H(a_1) \cdot [(c_1 + c) \mapsto E(id)] \\
(p, H, b^\bullet, E,\ id\ \texttt{=}\ id_1\ \ b) & \longmapsto & (p, H, b^\bullet, E \cdot [id \mapsto E(id_1)], b) \\
(p, H, b^\bullet, E,\ id\ \texttt{=}\ \texttt{alloc}\ \texttt{(}\ id_1\ \texttt{)}\ \ b) & \longmapsto & (p, H \cdot [a \mapsto t], b^\bullet, E \cdot [id \mapsto (a, 0)], b) \\
 & & \text{if } a \notin dom(H) \\
 & & \text{and } E(id_1) = c \wedge c \geq 0 \wedge c \text{ is divisible by } 4 \\
 & & \text{and } t = [0 \mapsto 0, 4 \mapsto 0, \ldots, (c-4) \mapsto 0] \\
(p, H, b^\bullet, E,\ \texttt{print}\ \texttt{(}\ id\ \texttt{)}\ \ b) & \longmapsto & (p, H, b^\bullet, E, b) \text{ and display } E(id) \\
(p, H, b^\bullet, E,\ \texttt{error}\ \texttt{(}\ s\ \texttt{)}\ \ b) & & \text{display } s \text{ and stop execution} \\
(p, H, b^\bullet, E,\ \texttt{goto}\ l\ \ b) & \longmapsto & (p, H, b^\bullet, E, b') \\
 & & \text{if } \mathsf{find}(b^\bullet, l) = b' \\
(p, H, b^\bullet, E,\ \texttt{if0}\ id\ \texttt{goto}\ l\ \ b) & \longmapsto & (p, H, b^\bullet, E, b') \\
 & & \text{if } E(id) = 0 \text{ and } \mathsf{find}(b^\bullet, l) = b' \\
(p, H, b^\bullet, E,\ \texttt{if0}\ id\ \texttt{goto}\ l\ \ b) & \longmapsto & (p, H, b^\bullet, E, b) \\
 & & \text{if } E(id) \neq 0
\end{array}
$$

$$
\frac{\begin{array}{c} E(id_0) = f \qquad p \text{ contains } \texttt{func}\ f\ \texttt{(}id'_1\ \ldots\ id'_f\texttt{)}\ b' \\ (p, H, b', [id'_1 \mapsto E(id_1), \ldots, id'_f \mapsto E(id_f)], b') \quad \longmapsto \quad (p, H', b', E', \texttt{return}\ id') \end{array}}{(p, H, b^\bullet, E,\ id\ \texttt{=}\ \texttt{call}\ id_0\ \texttt{(}id_1\ \ldots\ id_f\texttt{)}\ \ b) \quad \longmapsto \quad (p, H', b^\bullet, E \cdot [id \mapsto E'(id')], b)}
$$

4 Sparrow-V

4.1 Syntax

$$
\begin{array}{rrcl}
(\textit{Program}) & p & ::= & F_1 \ \ldots \ F_m \\
(\textit{FunDecl}) & F & ::= & \texttt{func} \ f \ \texttt{(}id_1 \ \texttt{...} \ id_f\texttt{)} \ b \\
(\textit{Block}) & b & ::= & i_1 \ \ldots \ i_n \ \texttt{return} \ id \\
(\textit{Instruction}) & i & ::= & l \texttt{:} \mid r \ \texttt{=} \ c \mid r \ \texttt{=} \ \texttt{@} \ f \\
& & \mid & r \ \texttt{=} \ r \ \texttt{+} \ r \mid r \ \texttt{=} \ r \ \texttt{-} \ r \mid r \ \texttt{=} \ r \ \texttt{*} \ r \mid r \ \texttt{=} \ r \ \texttt{<} \ r \\
& & \mid & r \ \texttt{=} \ \texttt{[} \ r \ \texttt{+} \ c \ \texttt{]} \mid \texttt{[} \ r \ \texttt{+} \ c \ \texttt{]} \ \texttt{=} \ r \mid r \ \texttt{=} \ r \mid id \ \texttt{=} \ r \mid r \ \texttt{=} \ id \\
& & \mid & r \ \texttt{=} \ \texttt{alloc} \ \texttt{(} \ r \ \texttt{)} \mid \texttt{print} \ \texttt{(} \ r \ \texttt{)} \mid \texttt{error} \ \texttt{(} \ s \ \texttt{)} \mid \texttt{goto} \ l \\
& & \mid & \texttt{if0} \ r \ \texttt{goto} \ l \mid r \ \texttt{=} \ \texttt{call} \ r \ \texttt{(}id \ \texttt{...} \ id\texttt{)} \\
(\textit{Register}) & r & ::= & \texttt{a2} \mid \texttt{a3} \mid \texttt{a4} \mid \texttt{a5} \mid \texttt{a6} \mid \texttt{a7} \\
& & \mid & \texttt{s1} \mid \texttt{s2} \mid \texttt{s3} \mid \texttt{s4} \mid \texttt{s5} \mid \texttt{s6} \mid \texttt{s7} \mid \texttt{s8} \mid \texttt{s9} \mid \texttt{s10} \mid \texttt{s11} \\
& & \mid & \texttt{t0} \mid \texttt{t1} \mid \texttt{t2} \mid \texttt{t3} \mid \texttt{t4} \mid \texttt{t5} \\
(\textit{FunctionName}) & f & ::= & \langle \textsf{FUNCTIONNAME} \rangle \\
(\textit{Label}) & l & ::= & \langle \textsf{LABEL} \rangle \\
(\textit{Identifier}) & id & ::= & \langle \textsf{IDENTIFIER} \rangle \setminus \{\texttt{a2}, \ldots, \texttt{a7}, \texttt{s1}, \ldots, \texttt{s11}, \texttt{t0}, \ldots, \texttt{t5}\} \\
(\textit{IntegerLiteral}) & c & ::= & \langle \textsf{INTEGER\_LITERAL} \rangle \\
(\textit{StringLiteral}) & s & ::= & \langle \textsf{STRING} \rangle
\end{array}
$$

We require that F_1 has no parameters. Also, in the instructions id `= [` id `+` c `]` and `[` id `+` c `] =` id, we require that $c \geq 0$ and that c is divisible by 4.

4.2 Helper Function

We define $\mathsf{find}(b, l) = b'$ such that $b = i_1 \ldots i_q \ \ l: \ b'$, that is, b' is the suffix of b that follows label l.

4.3 Values, Heaps, Environments, and Program States

Values. Sparrow-V has three kinds of values: integers c, heap addresses with offsets (a, c), and function names f. We use v to range over values.

The Heap. A heap H is a map from heap addresses a to tuples of values. A *heap address with offset* is a pair of the form (a, c), where a is a heap address and c is an integer such that $c \geq 0$ and c is divisible by 4.

The Register File. A register file R is a map from registers r to values.

The Environment. An environment E represents the parameters and local variables. An environment is a map from identifiers to values.

Program States. A program state $(p, H, R, b^{\bullet}, E, b)$ has six components. Intuitively, p is the entire program, H is the heap, R is the register file, $b^{\bullet}$ is the entire body of the function that is executing right now, E is an environment that represents the parameters and local variables, and b is the block that is executing right now.

The Initial Program State. Consider a program $p = F_1 \ \ldots \ F_m$ where $F_1 = \texttt{func} \ id \ \texttt{( )} \ b$. The initial program state is $(p, [\,], [\texttt{a0} \mapsto 0, \ldots \texttt{t6} \mapsto 0], b, [\,], b)$.

4.4 Semantics

The execution relation is the reflexive, transitive closure of $\longmapsto$, which we define as follows.

$$
\begin{array}{rcl}
(p,H,R,b^\bullet,E,\ l\texttt{:}\quad b) & \longmapsto & (p,H,R,b^\bullet,E,b) \\
(p,H,R,b^\bullet,E,\ r\ \texttt{=}\ c\quad b) & \longmapsto & (p,H,R\cdot[r\mapsto c],b^\bullet,E,b) \\
(p,H,R,b^\bullet,E,\ r\ \texttt{=}\ \texttt{@}\ f\quad b) & \longmapsto & (p,H,R\cdot[r\mapsto f],b^\bullet,E,b) \\
(p,H,R,b^\bullet,E,\ r\ \texttt{=}\ r_1\ \texttt{+}\ r_2\quad b) & \longmapsto & (p,H,R\cdot[r\mapsto(c_1+c_2)],b^\bullet,E,b) \\
 & & \text{if } R(r_1)=c_1\wedge R(r_2)=c_2 \\
(p,H,R,b^\bullet,E,\ r\ \texttt{=}\ r_1\ \texttt{+}\ r_2\quad b) & \longmapsto & (p,H,R\cdot[r\mapsto(a_1,c_1+c_2)],b^\bullet,E,b) \\
 & & \text{if } R(r_1)=(a_1,c_1)\wedge R(r_2)=c_2 \\
 & & \text{and } c_2\geq 0\wedge c_2 \text{ is divisible by } 4 \\
(p,H,R,b^\bullet,E,\ r\ \texttt{=}\ r_1\ \texttt{-}\ r_2\quad b) & \longmapsto & (p,H,R\cdot[r\mapsto(c_1-c_2)],b^\bullet,E,b) \\
 & & \text{if } R(r_1)=c_1\wedge R(r_2)=c_2 \\
(p,H,R,b^\bullet,E,\ r\ \texttt{=}\ r_1\ \texttt{*}\ r_2\quad b) & \longmapsto & (p,H,R\cdot[r\mapsto(c_1\times c_2),b^\bullet,E,b) \\
 & & \text{if } R(r_1)=c_1\wedge R(r_2)=c_2 \\
(p,H,R,b^\bullet,E,\ r\ \texttt{=}\ r_1\ \texttt{<}\ r_2\quad b) & \longmapsto & (p,H,R\cdot[r\mapsto 1],b^\bullet,E,b) \\
 & & \text{if } R(r_1)=c_1\wedge R(r_2)=c_2\wedge c_1<c_2 \\
(p,H,R,b^\bullet,E,\ r\ \texttt{=}\ r_1\ \texttt{<}\ r_2\quad b) & \longmapsto & (p,H,R\cdot[r\mapsto 0],b^\bullet,E,b) \\
 & & \text{if } R(r_1)=c_1\wedge R(r_2)=c_2\wedge c_1\geq c_2 \\
(p,H,R,b^\bullet,E,\ r\ \texttt{=}\ \texttt{[}\ r_1\ \texttt{+}\ c\ \texttt{]}\quad b) & \longmapsto & (p,H,R\cdot[r\mapsto(H(a_1))(c_1+c)],b^\bullet,E,b) \\
 & & \text{if } R(r_1)=(a_1,c_1)\wedge(c_1+c)\in \mathit{dom}(H(a_1)) \\
(p,H,R,b^\bullet,E,\ \texttt{[}\ r_1\ \texttt{+}\ c\ \texttt{]}\ \texttt{=}\ r\quad b) & \longmapsto & (p,H\cdot[a_1\mapsto t],R,b^\bullet,E,b) \\
 & & \text{if } R(r_1)=(a_1,c_1)\wedge(c_1+c)\in \mathit{dom}(H(a_1)) \\
 & & \text{and } t=H(a_1)\cdot[(c_1+c)\mapsto R(r)] \\
(p,H,R,b^\bullet,E,\ r\ \texttt{=}\ r_1\quad b) & \longmapsto & (p,H,R\cdot[r\mapsto R(r_1)],b^\bullet,E,b) \\
(p,H,R,b^\bullet,E,\ id\ \texttt{=}\ r\quad b) & \longmapsto & (p,H,R,b^\bullet,E\cdot[id\mapsto R(r)],b) \\
(p,H,R,b^\bullet,E,\ r\ \texttt{=}\ id\quad b) & \longmapsto & (p,H,R\cdot[r\mapsto E(id)],b^\bullet,E,b) \\
(p,H,R,b^\bullet,E,\ r\ \texttt{=}\ \texttt{alloc}\ \texttt{(}\ r_1\ \texttt{)}\quad b) & \longmapsto & (p,H\cdot[a\mapsto t],R\cdot[r\mapsto(a,0)],b^\bullet,E,b) \\
 & & \text{if } a\notin \mathit{dom}(H) \\
 & & \text{and } R(r_1)=c\wedge c\geq 0\wedge c \text{ is divisible by } 4 \\
 & & \text{and } t=[0\mapsto 0,4\mapsto 0,\ldots,(c-4)\mapsto 0] \\
(p,H,R,b^\bullet,E,\ \texttt{print}\ \texttt{(}\ r\ \texttt{)}\quad b) & \longmapsto & (p,H,R,b^\bullet,E,b) \text{ and display } R(r) \\
(p,H,R,b^\bullet,E,\ \texttt{error}\ \texttt{(}\ s\ \texttt{)}\quad b) & & \text{display } s \text{ and stop execution} \\
(p,H,R,b^\bullet,E,\ \texttt{goto}\ l\quad b) & \longmapsto & (p,H,R,b^\bullet,E,b') \\
 & & \text{if } \mathsf{find}(b^\bullet,l)=b' \\
(p,H,R,b^\bullet,E,\ \texttt{if0}\ r\ \texttt{goto}\ l\quad b) & \longmapsto & (p,H,R,b^\bullet,E,b') \\
 & & \text{if } R(r)=0 \text{ and } \mathsf{find}(b^\bullet,l)=b' \\
(p,H,R,b^\bullet,E,\ \texttt{if0}\ r\ \texttt{goto}\ l\quad b) & \longmapsto & (p,H,R,b^\bullet,E,b) \\
 & & \text{if } R(r)\neq 0
\end{array}
$$

$$
\frac{\begin{array}{c} E(r_0)=f \qquad p \text{ contains } \texttt{func}\ f\ \texttt{(}id'_1\ \ldots\ id'_f\texttt{)}\ b' \\ (p,H,R,b',[id'_1\mapsto E(id_1),\ldots,id'_f\mapsto E(id_f)],b') \quad\longmapsto\quad (p,H',R',b',E',\texttt{return}\ id') \end{array}}{(p,H,R,b^\bullet,E,\ r\ \texttt{=}\ \texttt{call}\ r_0\ \texttt{(}id_1\ \ldots\ id_f\texttt{)}\quad b) \quad\longmapsto\quad (p,H',R'\cdot[r\mapsto E'(id')],b^\bullet,E,b)}
$$