1 Syntax

$$
\begin{array}{lrcl}
\text{Instruction Fields} & f_i & ::= & f_{i_1} \mid \cdots \mid f_{i_k} \\
\text{Result Fields} & f_r & ::= & f_{r_1} \mid \cdots \mid f_{r_k} \\
\text{Fields} & f & ::= & f_i \mid f_r \\
\text{Instructions} & i & ::= & \{f_{i_1} = v_{i_1},\ \ldots\ ,\ f_{a_k} = v_{i_k}\} \\
\text{Results} & r & ::= & \{f_{r_1} = v_{r_1},\ \ldots\ ,\ f_{r_k} = v_{r_k}\}
\end{array}
$$

$$
\begin{array}{lrcll}
\text{Predicates} & a, b & ::= & 0 & \textit{Identity} \\
 & & \mid & 1 & \textit{False} \\
 & & \mid & f = n & \textit{Test} \\
 & & \mid & a + b & \textit{Sum} \\
 & & \mid & a \cdot b & \textit{Product} \\
 & & \mid & \neg\, a & \textit{Negation}
\end{array}
$$

$$
\begin{array}{lrcll}
\text{Policies} & p, q & ::= & a & \textit{Test} \\
 & & \mid & act(p) & \textit{Slice Actions} \\
 & & \mid & res(p) & \textit{Slice Results} \\
 & & \mid & inj_i & \textit{Injection Action} \\
 & & \mid & inj_r & \textit{Injection Result} \\
 & & \mid & f \leftarrow n & \textit{Update} \\
 & & \mid & p + q & \textit{Choice} \\
 & & \mid & p \cdot q & \textit{Sequential Concatenation} \\
 & & \mid & p^* & \textit{Kleene Star}
\end{array}
$$

2 Semantics

$$
\begin{aligned}
[\![\cdot]\!] \;&:\; P(A) \times P(R) \to P(A) \times P(R) \\
[\![0]\!](-,-) &\triangleq (\varnothing, \varnothing) \\
[\![1]\!](is, rs) &\triangleq (is, rs) \\
[\![f = n]\!](is, rs) &\triangleq (\mathsf{filter}\ (f = n)\ is, \mathsf{filter}\ (f = n)\ rs) \\
[\![a + b]\!](is, rs) &\triangleq \mathsf{let}\ (is_a, rs_a) = [\![a]\!](is, rs) \\
&\ \mathsf{let}\ (is_b, rs_b) = [\![b]\!](is, rs) \\
&\ (is_a \cup is_b, rs_a \cup rs_b) \\
[\![a \cdot b]\!](is, rs) &\triangleq \mathsf{let}\ (is_a, rs_a) = [\![a]\!](is, rs) \\
&\ \mathsf{let}\ (is_b, rs_b) = [\![b]\!](is, rs) \\
&\ (is_a \cap is_b, rs_a \cap rs_b) \\
[\![\neg a]\!](is, rs) &\triangleq \mathsf{let}\ (is', rs') = [\![a]\!](is, rs) \\
&\ (is - is', rs - rs') \\
[\![act(p)]\!](is, rs) &\triangleq \mathsf{let}\ (is', rs') = [\![p]\!](is, rs)\ \mathsf{in}\ (is', rs) \\
[\![res(p)]\!](is, rs) &\triangleq \mathsf{let}\ (is', rs') = [\![p]\!](is, rs)\ \mathsf{in}\ (is, rs') \\
[\![inj_i(i)]\!](is, rs) &\triangleq (\{i\} \cup is, rs) \\
[\![inj_r(r)]\!](is, rs) &\triangleq (is, \{r\} \cup rs) \\
[\![f \leftarrow n]\!](is, rs) &\triangleq (\mathsf{map}\ (f \leftarrow n)\ is, \mathsf{map}\ (f \leftarrow n)\ rs) \\
[\![p + q]\!](is, rs) &\triangleq \mathsf{let}\ (is_p, rs_p) = [\![p]\!](is, rs) \\
&\ \mathsf{let}\ (is_q, rs_q) = [\![q]\!](is, rs) \\
&\ (is_p \cup is_q, rs_p \cup rs_b) \\
[\![p \cdot q]\!](is, rs) &\triangleq \mathsf{let}\ (is', rs') = [\![p]\!](is, rs) \\
&\ [\![q]\!](is', rs') \\
&\cdots\cdots\cdots \\
[\![p^*]\!](is, rs) &\triangleq \mathsf{let}\ p^n = p_1 \cdot p_2 \cdot \ldots \cdot p_n \\
&\ \mathsf{let}\ (is_{p_n}, rs_{p_n}) = [\![p^n]\!](is, rs) \\
&\ (\bigcup_{n=0}^{\infty} is_{p_n}, \bigcup_{n=0}^{\infty} rs_{p_n})
\end{aligned}
$$

3 Applications

3.1 No Writes To R_0

$$\begin{array}{rcl} \textit{Instruction Fields } f_i & ::= & \mathsf{R}_{\mathsf{dest}} \\ \textit{Result Fields } f_r & ::= & (*\mathsf{empty}*) \\ \mathsf{WritesToZero} & \triangleq & \mathsf{R}_{\mathsf{dest}} = \mathsf{R}_0 \\ \mathsf{NoWritesToZero} & \triangleq & \mathit{act}(\neg\mathsf{WritesToZero}) \end{array}$$

Prove that

- if $[\![\mathsf{NoWritesZero}]\!](is, rs) = (is', rs')$
- then
  1. $rs = rs'$ and
  2. for all $i \in is'$, $i.\mathsf{R}_{\mathsf{dest}} \neq \mathsf{R}_0$.

3.2 Secure Memory Access

$$\begin{array}{rcl} \textit{Instruction Fields } f_i & ::= & \mathsf{PC} \\ \textit{Result Fields } f_r & ::= & \mathsf{ADDR} \\ \mathsf{SecureAddr} & \triangleq & \mathsf{ADDR} = 0 + \mathsf{ADDR} = 1 + \ldots + \mathsf{ADDR} = 100 \\ \mathsf{SecureInstr} & \triangleq & \mathsf{PC} = 0 + \mathsf{PC} = 1 + \ldots + \mathsf{PC} = 100 \\ \mathsf{SecureMemoryAccess} & \triangleq & \mathit{act}(\mathsf{SecureInstr}) + (\mathit{act}(\neg\mathsf{SecureInstr}) \cdot \mathit{res}(\neg\mathsf{SecureAddr})) \end{array}$$

Prove

- the results of secure instructions are unchanged, and
- the results of insecure instructions accessing insecure areas are unchanged.

3.3 Lock Bits

Each instruction is associated with a lock bit. Only the monitor can update lock bits. Certain operations like writing to a particular part of the address space require that the lock bit be set.