Cooma Expression Safety Proof

Language Definition

The language is defined by the following syntax:

$$
\begin{array}{rll}
b \in \text{boolean} & & \\
n \in \text{integer} & & \\
s \in \text{string} & & \\
v, x, y \in \text{identifiers} & & \\
\text{expression } e ::= & b & \text{boolean constant} \\
| & n & \text{integer constant} \\
| & s & \text{string constant} \\
| & x & \text{variable} \\
| & \{\} & \text{unit} \\
| & \text{fun}(x : t)\ e & \text{function abstraction} \\
| & e\ (e) & \text{function application} \\
| & \{\overline{x = e}\} & \text{record } (x_i \text{ distinct}) \\
| & e.x & \text{field selection} \\
| & < v = e > & \text{variant} \\
| & \Sigma\ r \text{ match } \overline{\text{case } y(x) \Rightarrow e} & \text{match} \\
 & & \\
\text{type } t ::= & t \rightarrow t & \text{function type} \\
| & \Pi\ r & \text{record type} \\
| & \Sigma\ r & \text{variant type} \\
| & b\ ! & \text{secret type} \\
| & b & \\
\text{base type } b ::= & \text{Boolean} & \text{boolean type} \\
| & \text{Int} & \text{integer type} \\
| & \text{String} & \text{string type} \\
| & \text{Unit} & \text{unit type} \\
\text{row } r ::= & \{\overline{x : t}\} & (x_i \text{ distinct})
\end{array}
$$

Computing Security Levels

Next we define a function, $\text{sec}(t)$, to compute the security value for a given type. This function should return a value which represents the highest security value used inside of the type. In this proof we only consider two levels of security; public (value of 0) and secret (value of 1). The function therefore returns a value of 1 for secret value and a value of 0 for public values if the type is a base value.

In the case of functions, we return the security level of the output value. If a function has a type $t_0 \rightarrow t_1$, then the typing rule states that $\text{sec}(t_0) \leq \text{sec}(t_1)$. The means that the security value for t_1, the output, must be at least as secure as the input value t_0. Therefore by returning the security value of t_1 we are guarenteed to be returning the highest security value for the given function.

Lastly, we consider the case of rows, which are used in record and variant types. To compute these values we use a helper function $\text{rsec}(r)$. The $\text{rsec}(\text{r})$ function takes a row and maps each type in the row to it's corresponding security value. It then takes the highest of all these security values as the security value for the whole row. This produces a value representing the most secure value in the row, which matches our inital description of the $\text{sec}(t)$ function; that it should return a value which represents the highest security value used inside of the type.

The corresponding definitions for $\text{sec}(t)$ and $\text{rsec}(r)$ are provided below:

$$
\begin{aligned}
&\text{sec}(t) : t \rightarrow \text{N} := \\
&\quad \mid t_0 \rightarrow t_1 \Rightarrow \text{sec}(t_1) \quad [\text{S-Fun}] \\
&\quad \mid \Pi r \Rightarrow \text{rsec}(r) \quad [\text{S-Rec}] \\
&\quad \mid \Sigma\, r \Rightarrow \text{rsec}(r) \quad [\text{S-Var}] \\
&\quad \mid b\ ! \Rightarrow 1 \quad [\text{S-Sec}] \\
&\quad \mid b \Rightarrow 0 \quad [\text{S-Pub}]
\end{aligned}
$$

$$
\text{rsec}(r) : r \rightarrow \text{N} := \max(\forall t \in r, t \rightarrow \text{sec}(t))
$$

We also define another helper function, $\text{sec-prop}(t)$, which is used to produce the required relationship between security values in a type. The function is used to catch types which violate the required security properties the top level instead of catching them at lower levels where type is defined. The function takes a type and returns a proposition, which provides the required relationship between types in a function. Since there is only one type in all other types, it returns True for these cases. It is defined as follows:

$$
\begin{aligned}
&\text{sec-prop}(t) : t \rightarrow \text{Prop} := \\
&\quad \mid t_0 \Rightarrow t_1 \Rightarrow \text{sec}(t_0) \leq \text{sec}(t_1) \wedge \text{sec-prop}(t_1) \quad [\text{TS-Fun}] \\
&\quad \mid \_ \Rightarrow \text{True} \quad [\text{TS-Oth}]
\end{aligned}
$$

Typing Rules

The typing rules for the language are defined as:

$$
\frac{}{\Gamma \vdash b \Uparrow \text{Boolean}}\ [\text{T-Boo}] \qquad \frac{}{\Gamma \vdash n \Uparrow \text{Int}}\ [\text{T-Int}] \qquad \frac{}{\Gamma \vdash s \Uparrow \text{String}}\ [\text{T-Str}]
$$

$$
\frac{}{\Gamma \vdash \{\} \Uparrow \text{Unit}}\ [\text{T-Uni}] \qquad \frac{x \in \Gamma \quad \text{sec-prop}(\Gamma(x))}{\Gamma \vdash x \Uparrow \Gamma(x)}\ [\text{T-Idn}]
$$

$$
\frac{\Gamma, x : t_0 \vdash e_0 \Uparrow t_1 \quad \text{sec-prop}(t_0 \rightarrow t_1)}{\Gamma \vdash \text{fun}(x : t_0)\ e_0 \Uparrow t_0 \rightarrow t_1}\ [\text{T-Fun}]
$$

$$
\frac{\Gamma \vdash e_0 \Uparrow t_0 \rightarrow t_1 \quad \Gamma \vdash e_1 \Downarrow t_0 \quad \text{sec-prop}(t_1)}{\Gamma \vdash e_0(e_1) \Uparrow t_1}\ [\text{T-App}]
$$

$$\frac{\Gamma \vdash \overline{e} \Uparrow \overline{t}}{\Gamma \vdash \{\overline{x = e}\} \Uparrow \Pi\ \{\overline{x : t}\}}\ [\text{T-Rec}] \qquad \frac{\Gamma \vdash e \Uparrow t}{\Gamma \vdash < v = e > \Uparrow \Sigma\ \{v : t\}}\ [\text{T-Var}]$$

$$\frac{\Gamma \vdash e \Uparrow \Pi\ \{\overline{x : t}\} \quad \text{sec-prop}(e.x_i)}{\Gamma \vdash e.x_i \Uparrow t_i}\ [\text{T-Sel}]$$

$$\frac{\forall i, \exists k, y_i = v_k \quad \Gamma, x_i : t_i \vdash e_0 \Uparrow t_0 \quad \text{sec-prop}(\Sigma\ \{\overline{v : t}\} \rightarrow t_0)}{\Gamma \vdash \Sigma\ \{\overline{v : t}\}\ \text{match}\ \overline{\text{case}\ y(x) \Rightarrow e_0} \Uparrow \Sigma\ \{\overline{v : t}\} \rightarrow t_0}\ [\text{T-Mat}]$$

$$\frac{\Gamma \vdash e \Uparrow u \quad u <: t}{\Gamma \vdash e \Downarrow t}\ [\text{T-Wid}]$$

Theorem (Expression Security)

We want to prove that we cannot produce a function which produces a value less secure then a given argument:

$$P = \nexists\, \Gamma \vdash e : t \rightarrow u \wedge \text{sec}(u) < \text{sec}(t)$$

We must prove that for every expression, e, P holds.

Boolean Constant

We prove that $P(e)$ holds when $e = b$:

$$P(b) = \nexists\, \Gamma \vdash b : t \rightarrow u \wedge \text{sec}(u) < \text{sec}(t)$$

We know from the $[\text{T-Boo}]$ rule that b can only have the type Boolean, which means we cannot have b of type $t \rightarrow u$.
This proves $P(e)$ holds when $e = b$.

Integer Constant

We prove that $P(e)$ holds when $e = n$:

$$P(n) = \nexists\, \Gamma \vdash n : t \rightarrow u \wedge \text{sec}(u) < \text{sec}(t)$$

We know from the $[\text{T-Int}]$ rule that n can only have the type Int, which means we cannot have n of type $t \rightarrow u$.
This proves $P(e)$ holds when $e = n$.

String Constant

We prove that $P(e)$ holds when $e = s$:

$$P(s) = \nexists\, \Gamma \vdash n : t \rightarrow u \wedge \text{sec}(u) < \text{sec}(t)$$

We know from the $[\text{T-Str}]$ rule that s can only have the type String, which means we cannot have s of type $t \rightarrow u$.
This proves $P(e)$ holds when $e = s$.

Unit

We prove that $P(e)$ holds when $e = \{\}$:

$$P(\{\}) = \nexists\, \Gamma \vdash \{\} : t \to u \wedge \text{sec}(u) < \text{sec}(t)$$

We know from the $[\text{T-Uni}]$ rule that $\{\}$ can only have the type Unit, which means we cannot have $\{\}$ of type $t \to u$.
This proves $P(e)$ holds when $e = \{\}$.

Identifiers

We prove that $P(e)$ holds when $e = x$:

$$P(x) = \nexists\, \Gamma \vdash x : t \to u \wedge \text{sec}(u) < \text{sec}(t)$$

By looking at the left side of the conjunction, we can apply the $[\text{T-Idn}]$ rule which tells us the following:

$$\begin{gathered}\Gamma(x) : t \to u \\ x \in \Gamma \\ \text{sec-prop}(\Gamma(x))\end{gathered}$$

By applying the defininton of sec-prop we get:

$$\begin{aligned}\text{sec-prop}(\Gamma(x)) &= \text{sec-prop}(t \to u) \\ &= \text{sec}(t) \leq \text{sec}(u) \wedge \text{sec-prop}(u)\end{aligned}$$

We can use the left side of the resulting conjunction $\text{sec}(t) \leq \text{sec}(u)$ to contradict the right side of our original conjunction P, that $\text{sec}(u) < \text{sec}(t)$.
This proves $P(e)$ holds when $e = x$.

Records

We prove that $P(e)$ holds when $e = \{\overline{x = e}\}$:

$$P(\{\overline{x = e}\}) = \nexists\, \Gamma \vdash \{\overline{x = e}\} : t \to u \wedge \text{sec}(u) < \text{sec}(t)$$

We know from the $[\text{T-Rec}]$ rule that $\{\overline{x = e}\}$ can only have the type $\Pi\ \{\overline{x : t}\}$, which means we cannot have $\{\overline{x = e}\}$ of type $t \to u$.
This proves $P(e)$ holds when $e = \{\overline{x = e}\}$.

Variants

We prove that $P(e)$ holds when $e =< v = e >$:

$$P(< v = e >) = \nexists\, \Gamma \vdash < v = e > : t \to u \wedge \text{sec}(u) < \text{sec}(t)$$

We know from the $[\text{T-Var}]$ rule that $< v = e >$ can only have the type $\Sigma\ \{v : t\}$, which means we cannot have $< v = e >$ of type $t \to u$.
This proves $P(e)$ holds when $e =< v = e >$.

Field Selection

We prove that $P(e)$ holds when $e = e.x$:

$$P(e.x) = \nexists\, \Gamma \vdash e.x : t \to u \wedge \text{sec}(u) < \text{sec}(t)$$

By looking at the left side of the conjunction, we can apply the [T-Sel] rule which tells us the following:

$$\begin{gathered} e.x_i : t \to u \\ e : \Pi\ \{\overline{x : t}\} \\ \text{sec-prop}(e.x_i) \end{gathered}$$

By applying the defininton of sec-prop we get:

$$\begin{aligned} \text{sec-prop}(e.x_i) &= \text{sec-prop}(t \to u) \\ &= \text{sec}(t) \leq \text{sec}(u) \wedge \text{sec-prop}(u) \end{aligned}$$

We can use the left side of the resulting conjunction $\text{sec}(t) \leq \text{sec}(u)$ to contradict the right side of our original conjunction P, that $\text{sec}(u) < \text{sec}(t)$.
This proves $P(e)$ holds when $e = e.x$.

Function Abstraction

We prove that $P(e)$ holds when $e = \text{fun}(x : t_0)\ e_0$

$$P(\text{fun}(x : t_0)\ e_0) = \nexists\, \Gamma \vdash \text{fun}(x : t_0)\ e_0 : t \to u \wedge \text{sec}(u) < \text{sec}(t)$$

By looking at the left side of the conjunction, we can apply the [T-Fun] rule which tells us the following:

$$\begin{gathered} t = t_0 \\ u = t_1 \\ e_0 : t_1 \\ \text{sec-prop}(t_0 \to t_1) \end{gathered}$$

By substituing the values for t and u into P we get:

$$\nexists\, \Gamma \vdash \text{fun}(x : t_0)\ e_0 : t_0 \to t_1 \wedge \text{sec}(t_1) < \text{sec}(t_0)$$

By applying the defininton of sec-prop we get:

$$\begin{aligned} \text{sec-prop}(t_0 \to t_1) &= \text{sec-prop}(t \to u) \\ &= \text{sec}(t) \leq \text{sec}(u) \wedge \text{sec-prop}(u) \end{aligned}$$

We can use the left side of the resulting conjunction $\text{sec}(t) \leq \text{sec}(u)$ to contradict the right side of our original conjunction P, that $\text{sec}(u) < \text{sec}(t)$.
This proves $P(e)$ holds when $e = \text{fun}(x : t_0)\ e_0$.

Function Application

We prove that $P(e)$ holds when $e = e_0(e_1)$:

$$P(e_0(e_1)) = \nexists\, \Gamma \vdash e_0(e_1) : t \to u \wedge \text{sec}(u) < \text{sec}(t)$$

By looking at the left side of the conjunction, we can apply the [T-App] rule which tells us the following:

$$t_1 = (t \to u)$$
$$e_0 : t_0 \to (t \to u)$$
$$e_1 : t_0$$
$$\text{sec-prop}(t_0 \to t_1)$$

By applying the defininton of sec-prop we get:

$$\begin{aligned}\text{sec-prop}(t_1) &= \text{sec-prop}(t \to u) \\ &= \text{sec}(t) \leq \text{sec}(u) \wedge \text{sec-prop}(u)\end{aligned}$$

We can use the left side of the resulting conjunction $\text{sec}(t) \leq \text{sec}(u)$ to contradict the right side of our original conjunction P, that $\text{sec}(u) < \text{sec}(t)$.
This proves $P(e)$ holds when $e = e_0(e_1)$.

Match

We prove that $P(e)$ holds when $e = \Sigma\ \{\overline{v : t}\}\ \text{match}\ \overline{\text{case}\ y(x) \Rightarrow e_0}$:

$$P(\Sigma\ \{\overline{v : t}\}\ \text{match}\ \overline{\text{case}\ y(x) \Rightarrow e_0}) = \nexists\ \Gamma \vdash \Sigma\ \{\overline{v : t}\}\ \text{match}\ \overline{\text{case}\ y(x) \Rightarrow e_0} : t \to u \wedge \text{sec}(u) < \text{sec}(t)$$

By looking at the left side of the conjunction, we can apply the $[\text{T-Mat}]$ rule which tells us the following:

$$t = \Sigma\ \{\overline{v : t}\}$$
$$u = t_0$$
$$\forall i, \exists k, y_i = v_k$$
$$e_0 : t_0$$
$$\text{sec-prop}(\Sigma\ \{\overline{v : t}\} \to t_0)$$

By substituting the values for t and u into P we get:

$$\nexists\ \Gamma \vdash \Sigma\ \{\overline{v : t}\}\ \text{match}\ \overline{\text{case}\ y(x) \Rightarrow e_0} : \Sigma\ \{\overline{v : t}\} \to t_0 \wedge \text{sec}(t_0) < \text{sec}(\Sigma\ \{\overline{v : t}\})$$

By applying the defininton of sec-prop we get:

$$\begin{aligned}\text{sec-prop}(\Sigma\ \{\overline{v : t}\} \to t_0) &= \text{sec-prop}(t \to u) \\ &= \text{sec}(t) \leq \text{sec}(u) \wedge \text{sec-prop}(u)\end{aligned}$$

We can use the left side of the resulting conjunction $\text{sec}(t) \leq \text{sec}(u)$ to contradict the right side of our original conjunction P, that $\text{sec}(u) < \text{sec}(t)$.
This proves $P(e)$ holds when $e = \Sigma\ \{\overline{v : t}\}\ \text{match}\ \overline{\text{case}\ y(x) \Rightarrow e_0}$.