EDA045 Program Analysis, 2018, Homework #5

December 5, 2018

Modalities

- **Deadline**: 18 December 2018, end of day
- **Submission mechanism**: Moodle
- **To submit**: Your results (pdf or plaintext), including the estimated number of hours you worked on the project

In the following you will be working on a variant of ATL (grammar: Figure 1, semantics: Figure 4, types: Figure 5), similar to the languages that we discussed in class.

If you find any obvious errors or unexpected behaviour in the semantics, note them down and use your best judgment to decide if the semantics need fixing. Make sure to justify any changes that you make to the original semantics. I will count such fixes so as to balance out mistakes elsewhere in this assignment.

Task #1: Extending ATL with one primitive (0.5 points)

Extend the ATL semantics above (which are based on the ones used in class) as follows:

- Introduce a new primitive, secret(⟨*expr*⟩), which (for now) simply returns its parameter. Add suitable rules to the operational semantics and the type system.

Task #2: Dynamic Taint Analysis for ATL (2.0 points)

Given the ATL language definition extended with the new primitive you added in Task #1, design a dynamic taint analysis that prevents the program from printing any value that has passed through secret(⟨*expr*⟩). For example, the line

```
print(secret(1))
```

must result in the program getting stuck without evaluating print.

$$
\begin{array}{lcl}
\mathit{name} & ::= & \mathit{id} \\
& | & \mathit{name}\ .\ \mathit{id} \\
\mathit{val} & ::= & \mathit{num} \\
& | & \mathsf{null} \\
\mathit{expr} & ::= & \langle \mathit{val} \rangle \\
& | & \langle \mathit{expr} \rangle \mathsf{+} \langle \mathit{expr} \rangle \\
& | & \langle \mathit{name} \rangle \\
& | & \mathsf{print}\ (\ \langle \mathit{val} \rangle\ ) \\
& | & \mathsf{new}\ (\ \mathit{id}\ ) \\
& | & \mathit{id}\ (\ \langle \mathit{expr} \rangle \star\ ) \\
\mathit{ty} & ::= & \mathsf{int} \\
& | & \mathit{id}
\end{array}
\qquad
\begin{array}{lcl}
\mathit{stmt} & ::= & \langle \mathit{name} \rangle\ \mathsf{=}\ \langle \mathit{expr} \rangle \\
& | & \{\ \langle \mathit{stmt} \rangle\ \} \\
& | & \mathsf{if}\ \langle \mathit{expr} \rangle\ \langle \mathit{stmt} \rangle\ \mathsf{else}\ \langle \mathit{stmt} \rangle \\
& | & \mathsf{while}\ \langle \mathit{expr} \rangle\ \langle \mathit{stmt} \rangle \\
& | & \mathsf{skip} \\
& | & \mathsf{return}\ \langle \mathit{expr} \rangle \\
& | & \langle \mathit{stmt} \rangle; \langle \mathit{stmt} \rangle \\
\mathit{decl} & ::= & \mathsf{proc}\ \mathit{id}\ (\ (\mathit{id} : \langle \mathit{ty} \rangle) \star\ )\ \langle \mathit{stmt} \rangle \\
& | & \mathsf{datatype}\ \mathit{id}(\ (\mathit{id} : \langle \mathit{ty} \rangle) \star\ ) \\
\mathit{prog} & ::= & \langle \mathit{decl} \rangle\ \langle \mathit{prog} \rangle \\
& | & \langle \mathit{stmt} \rangle
\end{array}
$$

Figure 1: ATL Grammar

$$
\begin{aligned}
p &\in \mathit{prog} \\
t, t_i &\in \mathit{ty} \\
\mathit{id}, \mathit{id}_i &\in \mathit{id} \\
n, n_i &\in \mathit{num} \\
v, v_i &\in \mathbb{V} = \mathit{num} \uplus \{\mathbf{null}\} \uplus \textsc{Ref} \uplus (\mathit{stmt} \times \mathit{id}\star) \\
e, e_i &\in \mathit{expr} \\
s, s_i &\in \mathit{stmt} \\
E, E', E'' &\in \mathbb{E} = \mathit{id} \rightharpoonup \mathbb{V} \\
\mathit{Ev} &\in \mathit{val} \uplus \mathbb{E} \\
H, H', H'', H''', H_i &\in \mathbb{H} = \textsc{Ref} \rightharpoonup \textsc{HObj} \\
r &\in \textsc{Ref} \\
\tau, \tau_i &\in \mathbb{T} = \{\textsc{Int}, \textsc{Null}, \textsc{Unit}\} \uplus \{T_{\mathit{id}} | \mathit{id} \in \mathit{id}\}
\end{aligned}
$$

Figure 2: Metavariables and their domains for later definitions.

$$\mathcal{N}(t) = \begin{cases} 0 & \iff \quad t = \mathsf{int} \\ \textbf{null} & \quad \text{otherwise} \end{cases} \qquad \mathcal{T}(t) = \begin{cases} \textsc{Int} & \iff \quad t = \mathsf{int} \\ T_t & \iff \quad t \in id \end{cases}$$

$$\frac{}{\langle \mathcal{D}_0, \mathcal{D}_T, E_p, s \rangle \blacktriangleright \langle \mathcal{D}_0, \mathcal{D}_T, E_p \rangle}$$

$$\frac{\langle \mathcal{D}_0, \mathcal{D}_T, E_p, p \rangle \blacktriangleright \langle \mathcal{D}'_0, \mathcal{D}'_T, E'_p \rangle \quad id \notin \text{dom } E'_p \quad E''_p = [id \mapsto \langle s, [id_1, \ldots, id_k] \rangle] E'_p}{\langle \mathcal{D}_0, \mathcal{D}_T, E_p, \textbf{proc } id(id_1 : t_1, \ldots, id_k : t_k)\ s\ p \rangle \blacktriangleright \langle \mathcal{D}'_0, \mathcal{D}'_T, E''_p \rangle}$$

$$\frac{\begin{array}{ll} id_i = id_k \iff i = k & \mathcal{D}''_0 = [id \mapsto [id_1 = \mathcal{N}(t_1), \ldots, id_k = \mathcal{N}(t_k)]] \mathcal{D}'_0 \\ \langle \mathcal{D}_0, \mathcal{D}_T, E_p, p \rangle \blacktriangleright \langle \mathcal{D}'_0, \mathcal{D}'_T, E'_p \rangle & \mathcal{D}''_T = [T_{id} \mapsto [id_1 : \mathcal{T}(t_1), \ldots, id_k : \mathcal{T}(t_k)]] \mathcal{D}'_T \end{array}}{\langle \mathcal{D}_0, \mathcal{D}_T, E_p, \textbf{datatype } id(id_1 : t_1, \ldots, id_k : t_k)\ p \rangle \blacktriangleright \langle \mathcal{D}''_0, \mathcal{D}''_T, E'_p \rangle}$$

Figure 3: ATL support definitions for deriving $\mathcal{D}_0$, $\mathcal{D}_T$, and E_p

On the other hand the following program should not result in the program getting stuck:

```
print(1)
```

- Adjust the operational semantics of the language to prevent any data that has been returned by secret from being printed.
  - Attempt to be maximally precise.
  - In your description, you can alter any rule in the operational semantics.
  - If your changes systematically affect a certain set of rules, you can describe the change abstractly. Make sure to be precise.
- Explain how and why your changes achieve the desired goal.

Task #3: Static Taint Analysis for ATL (2.5 points)

Given the ATL language definition extended with the new primitive you added in Task #1, design a static taint analysis that prevents the program from printing any value that has passed through secret(⟨*expr*⟩) (cf. Task #2 for an example).

- Either adjust the type system for ATL or introduce a new type system, approximating the dynamic analysis from Task #2. Your system should be nontrivial, e.g., handle the two examples listed for Task #1 with full precision.
- Explain any design tradeoffs that you have made.
- Give an example to demonstrate how your system is optimistic or conservative.

$$\frac{}{\langle H, E, v\rangle \Downarrow \langle H, v\rangle}\ (val) \qquad \frac{E(id) = v}{\langle H, E, id\rangle \Downarrow \langle H, v\rangle}\ (id)$$

$$\frac{\langle H, E, e_1\rangle \Downarrow \langle H', n_1\rangle \quad \langle H', E, e_2\rangle \Downarrow \langle H'', n_2\rangle \quad n = n_1 + n_2}{\langle E, e_1\texttt{+}e_2\rangle \Downarrow \langle H'', n\rangle}\ (plus)$$

$$\frac{E(id) = \langle s, [id_1, \ldots, id_k]\rangle \qquad \begin{array}{c}\langle H_0, E, e_1\rangle \Downarrow \langle H_1, v_1\rangle \cdots \langle H_{k-1}, E, e_k\rangle \Downarrow \langle H_k, v_k\rangle \\ \langle H_k, E_p \cup \{id_1 \mapsto v_1, \cdots, id_k \mapsto v_k\}, s\rangle \Downarrow \langle H', v\rangle\end{array}}{\langle H_0, E, \textsf{id}(e_1\ \ldots e_k)\rangle \Downarrow \langle H', v\rangle}\ (call)$$

$$\frac{\langle H, E, e\rangle \Downarrow \langle H', n\rangle}{\langle H, E, \textsf{print}(e)\rangle \Downarrow \langle H', 0\rangle}\ (print)$$

$$\frac{r\ fresh \quad \mathcal{D}_0(id) = [id_1 = v_1, \ldots, id_k = v_k]}{\langle H, E, \textsf{new (id)}\rangle \Downarrow \langle H \cup \{r \mapsto [id_1 \mapsto v_1, \ldots, id_k \mapsto v_k]\}, r\rangle}\ (new)$$

$$\frac{\langle H, E, e\rangle \Downarrow \langle H, r\rangle \quad H(r) = [id \mapsto v, \ldots]}{\langle H, E, e.id\rangle \Downarrow \langle H, v\rangle}\ (load)$$

$$\frac{\langle H, E, e_1\rangle \Downarrow \langle H', r\rangle \quad \langle H', E, e_2\rangle \Downarrow \langle H'', v\rangle \quad H''(\mathsf{r}) = R \quad R' = [id \mapsto v]R}{\langle H, E, e_1.id = e_2\rangle \Downarrow_s \langle [r \mapsto R']H'', E\rangle}\ (store)$$

$$\frac{\langle H, E, e\rangle \Downarrow \langle H', v\rangle}{\langle H, E, \textsf{id = e}\rangle \Downarrow_s \langle H', [id \mapsto v]E\rangle}\ (assign) \qquad \frac{\langle H, E, s\rangle \Downarrow_s \langle H', E'\rangle}{\langle H, E, \{\ s\ \}\rangle \Downarrow_s \langle H', E'\rangle}\ (block)$$

$$\frac{\langle H, E, s_1\rangle \Downarrow_s \langle H', E'\rangle \quad \langle H', E', s_2\rangle \Downarrow_s \langle H'', Ev\rangle}{\langle E, s_1;\ s_2\ \rangle \Downarrow_s \langle H'', Ev\rangle}\ (seq) \qquad \frac{}{\langle H, E, \textsf{skip}\rangle \Downarrow_s \langle H, E\rangle}\ (skip)$$

$$\frac{\langle H, E, e\rangle \Downarrow \langle H', v\rangle \quad v \in num \quad v \neq 0 \quad \langle H', E, s_1\rangle \Downarrow_s \langle H'', Ev\rangle}{\langle H, E, \textsf{if}\ e\ s_1\ \textsf{else}\ s_2\rangle \Downarrow_s \langle H'', Ev\rangle}\ (if\text{-}then)$$

$$\frac{\langle H, E, e\rangle \Downarrow \langle H', 0\rangle \quad \langle H', E, s_2\rangle \Downarrow_s \langle H'', Ev\rangle}{\langle H, E, \textsf{if}\ e\ s_1\ \textsf{else}\ s_2\rangle \Downarrow_s \langle H'', Ev\rangle}\ (if\text{-}else)$$

$$\frac{\langle H, E, e\rangle \Downarrow \langle H', 0\rangle}{\langle H, E, \textsf{while}\ e\ s\rangle \Downarrow_s \langle H', E\rangle}\ (while\text{-}done)$$

$$\frac{\begin{array}{ll}\langle H, E, e\rangle \Downarrow \langle H', v\rangle & \langle H', E, s\rangle \Downarrow_s \langle H'', E'\rangle \\ v \in num \qquad v \neq 0 & \langle H'', E', \textsf{while}\ e\ s\rangle \Downarrow_s \langle H''', E''\rangle\end{array}}{\langle H, E, \textsf{while}\ e\ s\rangle \Downarrow_s \langle H''', E''\rangle}\ (while\text{-}step)$$

$$\frac{\langle H, E, e\rangle \Downarrow \langle H', v\rangle}{\langle H, E, \textsf{return}\ e\rangle \Downarrow_s \langle H', v\rangle}\ (return)$$

Figure 4: ATL: natural semantics, assuming that $\mathcal{D}_0$ and E_p have been set up suitably (cf. Figure 3).

$$\frac{v \in num}{\Gamma \vdash v : \textsc{Int}}\ (t\text{-}num) \qquad \frac{}{\Gamma \vdash \textbf{null} : \textsc{Null}}\ (t\text{-}null) \qquad \frac{}{\Gamma \cup \{id : \tau\} \vdash id : \tau}\ (t\text{-}id)$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \cdots \quad \Gamma \vdash e_k : \tau_k \quad \Gamma \vdash F_{id} : \tau_1 \times \cdots \times \tau_k \to \tau_0}{\Gamma \vdash id(e_1, \ldots, e_k) : \tau_0}\ (t\text{-}call)$$

$$\frac{\Gamma \vdash e_1 : \textsc{Int} \quad \Gamma \vdash e_2 : \textsc{Int}}{\Gamma \vdash e_1 + e_2 : \textsc{Int}}\ (t\text{-}plus) \qquad \frac{\Gamma \vdash e : \textsc{Int}}{\Gamma \vdash \textbf{print}(e) : \textsc{Int}}\ (t\text{-}print)$$

$$\frac{id \in \mathrm{dom}\ \mathcal{D}_T}{\Gamma \vdash \textbf{new}(id) : T_{id}}\ (t\text{-}new) \qquad \frac{\Gamma \vdash e : T_{id} \quad \mathcal{D}_T(id) = [id_1 : \tau, \ldots]}{\Gamma \vdash e.id_1 : \tau}\ (t\text{-}load)$$

$$\frac{\Gamma \vdash e_1 : T_{id} \quad \mathcal{D}_T(id) = [id_1 : \tau, \ldots] \quad \Gamma \vdash e_2 \triangleleft \tau}{\Gamma \vdash_s e_1.id_1 = e_2 : \textsc{Unit}}\ (t\text{-}store)$$

$$\frac{\Gamma \vdash id : \tau \quad \Gamma \vdash e \triangleleft \tau}{\Gamma \vdash_s id = e : \textsc{Unit}}\ (t\text{-}assign) \qquad \frac{\Gamma \vdash_s s : \tau}{\Gamma \vdash_s \{\ s\ \} : \tau}\ (t\text{-}block)$$

$$\frac{\Gamma \vdash s_1 : \textsc{Unit} \quad \Gamma \vdash s_2 : \tau}{\Gamma \vdash_s s_1; s_2 : \tau}\ (t\text{-}seq) \qquad \frac{}{\Gamma \vdash_s \textbf{skip} : \textsc{Unit}}\ (t\text{-}skip)$$

$$\frac{\Gamma \vdash e_1 : \textsc{Int} \quad \Gamma \vdash_s s_1 : \tau \quad \Gamma \vdash_s s_2 : \tau}{\Gamma \vdash_s \textbf{if}\ e_1\ s_1\ \textbf{else}\ s_2 : \tau}\ (t\text{-}if)$$

$$\frac{\Gamma \vdash e_1 : \textsc{Int} \quad \Gamma \vdash_s s : \textsc{Unit}}{\Gamma \vdash_s \textbf{while}\ e\ s : \textsc{Unit}}\ (t\text{-}while) \qquad \frac{\Gamma \vdash e : \tau}{\Gamma \vdash_s \textbf{return}\ e : \tau}\ (t\text{-}return)$$

$$\frac{\begin{array}{l}\tau_1 = \mathcal{T}(t_1) \quad \cdots \quad \tau_k = \mathcal{T}(t_k) \\ F_{id} : \tau_1 \times \cdots \times \tau_k \to \tau_0 \in \Gamma\end{array} \qquad \begin{array}{r}\Gamma \cup \{id_1 : \tau_1, \ldots, id_k : \tau_k\} \vdash_s s : \tau_0 \\ \Gamma \vdash_p p : \tau\end{array}}{\Gamma \vdash_s \textbf{proc}\ id(id_1 : t_1, \ldots, id_k : t_k)\ s\ p : \tau}\ (t\text{-}proc)$$

$$\frac{\Gamma \vdash_s p : \tau}{\Gamma \vdash_s \textbf{datatype}\ id(id_1 : t_1, \ldots, id_k : t_k)\ p : \tau}\ (t\text{-}dataty)$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash e \triangleleft \tau}\ (assignable\text{-}ty) \qquad \frac{\Gamma \vdash e : \textsc{Null}}{\Gamma \vdash e \triangleleft T_{id}}\ (assignable\text{-}null)$$

Figure 5: ATL: type inference rules (monomorphic), assuming that $\mathcal{D}_T$ has been set up suitably (cf. Figure 3).