Lecture: Program analysis
Exercise 7
http://proglang.informatik.uni-freiburg.de/teaching/programanalysis/2010ss/

1 Control Flow Analysis for an object-oriented language

Program	::=	Class* Exp
Class	::=	**class** Id Var* Method* **end**
Var	::=	**var** Id
Method	::=	**method** Id (Id*) Exp **end**
Exp	::=	Term^l
Term	::=	Int | Id | Exp Op Exp | **false** | **true** | Id := Exp |
		if Exp **then** Exp **else** Exp **end** | Exp;Exp |
		this | **null** | **new** Id | Exp.Id(Exp*)
Op	::=	$+ \mid - \mid * \mid \& \mid < \mid =$
Id	::=	⟨identifier⟩
Int	::=	⟨integer⟩

Consider the object-oriented mini-language defined above. It implements standard semantics, assuming the following rules:

- All variables are initialized with **null**.
- Assignments evaluate to the expression on the right-hand side.
- You may assume that all instance variables and formal arguments have distinct names. Further, **this** is never used outside classes; when used within a class C, it is renamed to **this-C**.

Define a 0-CFA for this language which determines for each expression to elements of which type(s) it might evaluate. Possible types are **Bool**, **Int**, and $C \in \mathbf{CName}_*$, where $\mathbf{CName}_*$ is the set of all classes defined in a program.

1. What are $C(l)$ and $r(x)$ in this setting?
2. Define for each kind of expression the set of constraints $\mathcal{C}_*$ it generates.
3. Consider the following type-incorrect program:

```
class C
   method n(i)
     i+1
   end
end

(new C).n(true)
```

   Add labels and give the constraints that are generated for this program together with a minimal solution.
4. How can the results of the 0-CFA be used to reject programs which are not type-correct?

Solution

1. We define $\mathcal{T} = \{\mathbf{Int}, \mathbf{Bool}\} \cup \mathbf{CName}_*$ and $r : \mathbf{Var}_* \to \mathcal{P}(\mathcal{T},)\ C : \mathbf{Lab}_* \to \mathcal{P}(\mathcal{T})$.
2. The constraints could be defined as follows:

$$
\begin{array}{lll}
[int] & \mathcal{C}_*[\![i^l]\!] & = \{\{\textbf{Int}\} \subseteq C(l)\} \\
[true] & \mathcal{C}_*[\![\textbf{true}^l]\!] & = \{\{\textbf{Bool}\} \subseteq C(l)\} \\
[false] & \mathcal{C}_*[\![\textbf{false}^l]\!] & = \{\{\textbf{Bool}\} \subseteq C(l)\} \\
[op+] & \mathcal{C}_*[\![[e_1 + e_2]^l]\!] & = \mathcal{C}_*[\![e_1]\!] \cup \mathcal{C}_*[\![e_2]\!] \cup \{\{\textbf{Int}\} \subseteq C(l)\} \\
[ass] & \mathcal{C}_*[\![[x := t^{l_0}]^l]\!] & = \mathcal{C}_*[\![t^{l_0}]\!] \cup \{C(l_0) \subseteq r(x)\} \cup \{C(l_0) \subseteq C(l)\} \\
[if] & \mathcal{C}_*[\![[\textbf{if}\, t_0^{l_0}\, \textbf{then}\, t_1^{l_1}\, \textbf{else}\, t_2^{l_2}\, \textbf{end}]^l]\!] & = \mathcal{C}_*[\![t_0^{l_0}]\!] \cup \mathcal{C}_*[\![t_1^{l_1}]\!] \cup \mathcal{C}_*[\![t_2^{l_2}]\!] \\
 & & \quad \cup \{C(l_1) \subseteq C(l)\} \cup \{C(l_2) \subseteq C(l)\} \\
[seq] & \mathcal{C}_*[\![[t_1^{l_1}; t_2^{l_2}]^l]\!] & = \mathcal{C}_*[\![t_1^{l_1}]\!] \cup \mathcal{C}_*[\![t_2^{l_2}]\!] \cup \{C(l_2) \subseteq C(l)\} \\
[this] & \mathcal{C}_*[\![\textbf{this} - \textbf{C}^l]\!] & = \{\{C\} \subseteq C(l)\} \\
[new] & \mathcal{C}_*[\![[\textbf{new}\, C]^l]\!] & = \{\{C\} \subseteq C(l)\} \\
[var] & \mathcal{C}_*[\![x^l]\!] & = \{r(x) \subseteq C(l)\} \\
[null] & \mathcal{C}_*[\![\textbf{null}^l]\!] & = \emptyset \\
[call] & \mathcal{C}_*[\![[(t_0^{l_0}).m(t_1^{l_1}, \ldots, t_n^{l_n})]^l]\!] & = \bigcup_{i=0}^{n} \mathcal{C}_*[\![t_i^{l_i}]\!] \\
 & & \cup \{\{C\} \subseteq C(l_0) \Rightarrow C(l_i) \subseteq r(x_i)\, \forall i = 1 \ldots n \,| \\
 & & \qquad C \text{ defines } \textbf{method}\, m\,(x_1, \ldots, x_n)\, t_m^m\, \textbf{end}\} \\
 & & \cup \{\{C\} \subseteq C(l_0) \Rightarrow C(l_m) \subseteq C(l) | \\
 & & \qquad C \text{ defines } \textbf{method}\, m\,(x_1, \ldots, x_n)\, t_m^m\, \textbf{end}\}
\end{array}
$$

Similarly for all other binary operators.

3. The labeled program could look like this:

```
class C
    method n(i)
        (i^1 + 1^2)^3
    end
end

((new C)^4.n(true^5))^6
```

The constraints for this program are

$$
\begin{array}{rcl}
r(i) & \subseteq & C(1) \\
\{\textbf{Int}\} & \subseteq & C(2) \\
\{\textbf{Int}\} & \subseteq & C(3) \\
\{C\} & \subseteq & C(4) \\
\{\textbf{Bool}\} & \subseteq & C(5) \\
C \in C(4) & \Rightarrow & C(5) \subseteq r(i) \\
C \in C(4) & \Rightarrow & C(3) \subseteq C(6)
\end{array}
$$

A minimal solution is given by:

$$
\begin{array}{rcl}
C(1) & = & \{\textbf{Bool}\} \\
C(2) & = & \{\textbf{Int}\} \\
C(3) & = & \{\textbf{Int}\} \\
C(4) & = & \{C\} \\
C(5) & = & \{\textbf{Bool}\} \\
C(6) & = & \{\textbf{Int}\} \\
r(i) & = & \{\textbf{Bool}\}
\end{array}
$$

4. If we annotate the program with the inferred type information, we could run a type checker. The type checker would then detect the type error in the sum.

2 Correctness of 0-CFA

1. The following statement was crucial in the correctness proof for 0-CFA (cf. Slide 47 or Fact 3.11 on p. 160):

$$\Big((\widehat{C}, \widehat{\rho}) \models it^{l_1} \;\wedge\; \widehat{C}(l_1) \subseteq \widehat{C}(l_2)\Big) \quad \Rightarrow \quad (\widehat{C}, \widehat{\rho}) \models it^{l_2} \tag{1}$$

Prove the statement formally.

2. Reconsider the decision to use $\widehat{\mathbf{Val}} = \mathcal{P}(\mathbf{Term})$ in the correctness proof. Alternatively, we could have chosen $\widehat{\mathbf{Val}} = \mathcal{P}(\mathbf{Exp})$. Show that the specification of the CFA may be modified accordingly, but that then the statement 1 above (and hence the correctness result) would fail.

Solution

1. Proof by each case.

$$
\begin{array}{rl}
[con] & (\widehat{C}, \widehat{\rho}) \models c^{l_1} \quad \text{always} \Rightarrow (\widehat{C}, \widehat{\rho}) \models c^{l_2} \\
[var] & (\widehat{C}, \widehat{\rho}) \models x^{l_1} \ \wedge \ \widehat{C}(l_1) \subseteq \widehat{C}(l_2) \Leftrightarrow \rho(x) \subseteq \widehat{C}(l_1) \subseteq \widehat{C}(l_2) \\
& \Rightarrow (\widehat{C}, \widehat{\rho}) \models x^{l_2} \\
[fn] & (\widehat{C}, \widehat{\rho}) \models (\mathbf{fn}\, x \Rightarrow e_0)^{l_1} \ \wedge \ \widehat{C}(l_1) \subseteq \widehat{C}(l_2) \Leftrightarrow (\mathbf{fn}\, x \Rightarrow e_0) \subseteq \widehat{C}(l_1) \subseteq \widehat{C}(l_2) \\
& \Rightarrow (\widehat{C}, \widehat{\rho}) \models (\mathbf{fn}\, x \Rightarrow e_0)^{l_2}
\end{array}
$$

All other cases proceed similarly.

2. Define $\widehat{v} \in \widehat{Val} = \mathcal{P}(\mathbf{Exp})$.

$$
\begin{array}{rl}
[con] & (\widehat{C}, \widehat{\rho}) \models c^{l} \quad \text{always} \\
[var] & (\widehat{C}, \widehat{\rho}) \models x^{l} \text{ iff } \widehat{\rho}(x) \subseteq \widehat{C}(l) \\
[fn] & (\widehat{C}, \widehat{\rho}) \models (\mathbf{fn}\, x \Rightarrow e_0)^{l} \text{ iff } \{(\mathbf{fn}\, x \Rightarrow e_0)^{l}\} \subseteq \widehat{C}(l) \\
[fun] & (\widehat{C}, \widehat{\rho}) \models (\mathbf{fun}\, f\, x \Rightarrow e_0)^{l} \text{ iff } \{(\mathbf{fun}\, f\, x \Rightarrow e_0)^{l}\} \subseteq \widehat{C}(l) \\
[app] & (\widehat{C}, \widehat{\rho}) \models (t_1^{l_1}\ t_2^{l_2})^{l} \text{ iff } (\widehat{C}, \widehat{\rho}) \models t_1^{l_1} \ \wedge \ (\widehat{C}, \widehat{\rho}) \models t_2^{l_2} \\
& \wedge \big(\forall (\mathbf{fn}\, x \Rightarrow t_0^{l_0})^{l_3} \in \widehat{C}(l_1) : (\widehat{C}, \widehat{\rho}) \models t_0^{l_0} \ \wedge \ \widehat{C}(l_2) \subseteq \widehat{\rho}(x) \ \wedge \ \widehat{C}(l_0) \subseteq \widehat{C}(l)\big) \\
& \wedge \big(\forall (\mathbf{fun}\, f\, x \Rightarrow t_0^{l_0})^{l_3} \in \widehat{C}(l_1) : (\widehat{C}, \widehat{\rho}) \models t_0^{l_0} \ \wedge \ \widehat{C}(l_2) \subseteq \widehat{\rho}(x) \ \wedge \ \widehat{C}(l_0) \subseteq \widehat{C}(l) \\
& \qquad \wedge \ (\mathbf{fun}\, f\, x \Rightarrow t_0^{l_0})^{l_3} \subseteq \widehat{\rho}(f)\big)
\end{array}
$$

All other rules remain unchanged.

For an example where statement 1 fails consider $it = (\mathbf{fn}\, x \Rightarrow e_0)$, and $ie_1 = it^{l_1}, ie_2 = it^{l_2}$. Assume, that $(\widehat{C}, \widehat{\rho}) \models ie_1$, i.e. $\{(\mathbf{fn}\, x \Rightarrow e_0)^{l_1}\} \subseteq \widehat{C}(l_1)$. Now, choose $\widehat{C}(l_1) = \widehat{C}(l_2) = \{(\mathbf{fn}\, x \Rightarrow e_0)^{l_1}\}$. Then, the condition of the statement holds but $(\widehat{C}, \widehat{\rho}) \models ie_2$ does not hold because $\{ie_2\} \notin \widehat{C}(l_2)$.