Interprocedural Version of the Reaching Definitions Analysis

This is based on the version typeset by Kristina Boysen, with corrections from Jing Liu, Ru He, Cui Ye and Neeraj Khanolkar, Samik Basu, and ultimately on the book *Principles of Programming Analysis* [1].

The property space for the intraprocedural version is $L = \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star^?)$, so the property space for the interprocedural analysis is as follows.

$$\begin{array}{l}\widehat{L} = \Delta \to \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star^?)\\ \Delta = \mathbf{Lab}^*\end{array}$$

The dataflow equations in this case are as follows (recall that $S_\star$ is the main statement, whereas $P_\star$ is the entire program, including $D_\star$).

$$\begin{array}{lcl}\mathsf{RD}_{entry}(\ell)(\delta) & = & \begin{cases}\{(x,?) \mid x \in FV(S_\star)\}, & \text{if } \delta = [\,] \wedge \ell \in init(S_\star)\\ \bigcup\{\mathsf{RD}_{exit}(\ell')(\delta) \mid \ (\ell',\ell) \in flow(P_\star)\}, & \text{otherwise}\end{cases}\\ \mathsf{RD}_{exit}(\ell)(\delta) & = & \begin{cases}\widehat{f^1_\ell}(\mathsf{RD}_{entry}(\ell))(pop(\delta)), & \text{if } call(\ell)\\ \widehat{f^2_{\ell_c,\ell}}(\mathsf{RD}_{entry}(\ell_c),\mathsf{RD}_{entry}(\ell))(\delta), & \text{if } return(\ell_c,\ell)\\ \widehat{f_\ell}(\mathsf{RD}_{entry}(\ell))(\delta), & \text{otherwise}\end{cases}\end{array}$$

where

$$\begin{array}{lcl}\widehat{f_\ell}(Y)(\delta) & = & (Y(\delta) \setminus kill_{\mathsf{RD}}(B^\ell)) \cup gen_{\mathsf{RD}}(B^\ell)\\ & & \text{where } B^\ell \in \mathbf{Blocks}_\star\\ \widehat{f^1_{\ell_c}}(Y)(\delta) & = & (Y(\delta) \setminus \{(x,\ell') \mid x \in formals(\ell_c), \ell' \in \mathbf{Lab}_\star^?\})\\ & & \cup\{(x,\ell_c) \mid x \in val\text{-}formals(\ell_c)\}\\ & & \cup\{(y,?) \mid y \in res\text{-}formals(\ell_c)\}\\ \widehat{f^2_{\ell_c,\ell_r}}(X,Y)(\delta) & = & ((\{(x,\ell) \mid (x,\ell) \in X(\delta), x \in formals(\ell_r), \ell \in \mathbf{Lab}_\star^?\}\\ & & \quad \cup (Y(\delta : \ell_c) \setminus \{(x,\ell') \mid x \in formals(\ell_r), \ell' \in \mathbf{Lab}_\star^?\}))\\ & & \quad \setminus \{(z,\ell') \mid z \in res\text{-}actuals(\ell_r), \ell' \in \mathbf{Lab}_\star^?\})\\ & & \cup \{(z,\ell_r) \mid z \in res\text{-}actuals(\ell_r)\}\\ pop(\delta : \ell) & = & \delta\\ call(\ell) & = & (\exists \ell_x,\ell_n,\ell_r \in \mathbf{Lab}_\star :: (\ell,\ell_n,\ell_x,\ell_r) \in inter\text{-}flow_\star)\\ return(\ell_c,\ell) & = & (\exists \ell_x,\ell_n \in \mathbf{Lab}_\star :: (\ell_c,\ell_n,\ell_x,\ell) \in inter\text{-}flow_\star)\\ formals(\ell) & = & val\text{-}formals(\ell) \cup res\text{-}formals(\ell)\\ val\text{-}formals(\ell) & = & \{x \mid calledAt(\ell,p), \texttt{proc}\ p(\texttt{val}\ x, \texttt{res}\ y)\ \texttt{is}^{\ell_n}\ S\ \texttt{end}^{\ell_x} \in D_\star\}\\ res\text{-}formals(\ell) & = & \{y \mid calledAt(\ell,p), \texttt{proc}\ p(\texttt{val}\ x, \texttt{res}\ y)\ \texttt{is}^{\ell_n}\ S\ \texttt{end}^{\ell_x} \in D_\star\}\\ res\text{-}actuals(\ell) & = & \{z \mid [\texttt{call}\ p(a,z)]^\ell_{\ell_r} \in \mathbf{Blocks}_\star \vee [\texttt{call}\ p(a,z)]^{\ell_c}_\ell \in \mathbf{Blocks}_\star\}\\ calledAt(\ell,p) & = & [\texttt{call}\ p(a,z)]^\ell_{\ell_r} \in \mathbf{Blocks}_\star \vee [\texttt{call}\ p(a,z)]^{\ell_c}_\ell \in \mathbf{Blocks}_\star\end{array}$$

Note that $kill_{\mathsf{RD}}$ and gen_{RD} have to be extended to `is` and `end` blocks as follows. (There is no need to extend the definiton to `call` blocks, because these functions are not used for such blocks.)

$$\begin{array}{lcl}kill_{\mathsf{RD}}(\texttt{is}^\ell) & = & \{\}\\ kill_{\mathsf{RD}}(\texttt{end}^\ell) & = & \{\}\\ gen_{\mathsf{RD}}(\texttt{is}^\ell) & = & \{\}\\ gen_{\mathsf{RD}}(\texttt{end}^\ell) & = & \{\}\end{array}$$

Neeraj pointed out some problems with scoping rules may still remain. There may be several ways to fix these problems, however, it seems that one way is to assume that the formal parameters

in all procedures are distinct (across procedures). To keep things relatively simple, we'll make that assumption.

After many corrections, it seems that the only way to be sure this analysis is right is to do a correctness proof. But short of doing that, it helps to work some examples.

Here's a simple example:

```
begin
  proc p(val n, res r) is[1]
    [r := n][2]
  end[3]
  [y := n][4]
  [call p(y, q)]5[6]
end
```

According to my informal understanding of Reaching Definitions, this program should have the following solution to its dataflow equations.

$$
\begin{array}{lcl}
\mathsf{RD}_{entry}(4)([]) & = & \{(\mathtt{n},?),(\mathtt{q},?),(\mathtt{y},?)\} \\
\mathsf{RD}_{exit}(4)([]) & = & \{(\mathtt{n},?),(\mathtt{q},?),(\mathtt{y},4)\} \\
\mathsf{RD}_{entry}(5)([]) & = & \{(\mathtt{n},?),(\mathtt{q},?),(\mathtt{y},4)\} \\
\mathsf{RD}_{exit}(5)([5]) & = & \{(\mathtt{n},5),(\mathtt{q},?),(\mathtt{r},?),(\mathtt{y},4)\} \\
\mathsf{RD}_{entry}(1)([5]) & = & \{(\mathtt{n},5),(\mathtt{q},?),(\mathtt{r},?),(\mathtt{y},4)\} \\
\mathsf{RD}_{exit}(1)([5]) & = & \{(\mathtt{n},5),(\mathtt{q},?),(\mathtt{r},?),(\mathtt{y},4)\} \\
\mathsf{RD}_{entry}(2)([5]) & = & \{(\mathtt{n},5),(\mathtt{q},?),(\mathtt{r},?),(\mathtt{y},4)\} \\
\mathsf{RD}_{exit}(2)([5]) & = & \{(\mathtt{n},5),(\mathtt{q},?),(\mathtt{r},2),(\mathtt{y},4)\} \\
\mathsf{RD}_{entry}(3)([5]) & = & \{(\mathtt{n},5),(\mathtt{q},?),(\mathtt{r},2),(\mathtt{y},4)\} \\
\mathsf{RD}_{exit}(3)([5]) & = & \{(\mathtt{n},5),(\mathtt{q},?),(\mathtt{r},2),(\mathtt{y},4)\} \\
\mathsf{RD}_{entry}(6)([5]) & = & \{(\mathtt{n},5),(\mathtt{q},?),(\mathtt{r},2),(\mathtt{y},4)\} \\
\mathsf{RD}_{exit}(6)([]) & = & \{(\mathtt{n},?),(\mathtt{q},6),(\mathtt{y},4)\}
\end{array}
$$

The crucial parts of the above are the computations of $\mathsf{RD}_{exit}(5)([5])$ and $\mathsf{RD}_{exit}(6)([])$. To check these we calculate as follows.

First, assume that $\mathsf{RD}_{entry}(5)([])$ is $\{(\mathtt{n},?),(\mathtt{q},?),(\mathtt{y},4)\}$. Then we can calculate $\mathsf{RD}_{exit}(5)([5])$ as follows.

$$
\begin{array}{ll}
 & \mathsf{RD}_{exit}(5)([5]) \\
= & \quad \langle \text{by definition, since } \mathit{call}(5) \text{ is true} \rangle \\
 & \widehat{f_5^1}(\mathsf{RD}_{entry}(5))(\mathit{pop}([5])) \\
= & \quad \langle \text{by definition of } \mathit{pop} \rangle \\
 & \widehat{f_5^1}(\mathsf{RD}_{entry}(5))([]) \\
= & \quad \langle \text{by definition of } \widehat{f_5^1} \rangle \\
 & (\mathsf{RD}_{entry}(5)([]) \setminus \{(x,\ell') \mid x \in \mathit{formals}(5), \ell' \in \mathbf{Lab}_\star^?\}) \\
 & \cup \{(x,5) \mid x \in \mathit{val\text{-}formals}(5)\} \\
 & \cup \{(y,?) \mid y \in \mathit{res\text{-}formals}(5)\} \\
= & \quad \langle \text{by the program and definitions of auxiliary functions} \rangle \\
 & (\mathsf{RD}_{entry}(5)([]) \setminus \{(x,\ell') \mid x \in \{\mathtt{n},\mathtt{r}\}, \ell' \in \{1,2,3,4,5,6\}\}) \\
 & \cup \{(x,5) \mid x \in \{\mathtt{n}\}\} \\
 & \cup \{(y,?) \mid y \in \{\mathtt{r}\}\} \\
= & \quad \langle \text{by assumption that } \mathsf{RD}_{entry}(5)([]) \text{ is } \{(\mathtt{n},?),(\mathtt{q},?),(\mathtt{y},4)\} \rangle \\
 & (\{(\mathtt{n},?),(\mathtt{q},?),(\mathtt{y},4)\} \setminus \{(x,\ell') \mid x \in \{\mathtt{n},\mathtt{r}\}, \ell' \in \{1,2,3,4,5,6\}\}) \\
 & \cup \{(x,5) \mid x \in \{\mathtt{n}\}\} \\
 & \cup \{(y,?) \mid y \in \{\mathtt{r}\}\} \\
= & \quad \langle \text{by set theory} \rangle
\end{array}
$$

$$
\begin{array}{ll}
 & (\{(\mathtt{q},?),(\mathtt{y},4)\} \cup \{(x,5) \mid x \in \{\mathtt{n}\}\} \cup \{(y,?) \mid y \in \{\mathtt{r}\}\} \\
= & \quad \langle \text{by set theory} \rangle \\
 & \{(\mathtt{n},5),(\mathtt{q},?),(\mathtt{r},?),(\mathtt{y},4)\}
\end{array}
$$

Now, assume that $\mathsf{RD}_{entry}(6)([5])$ is $\{(\mathtt{n},5),(\mathtt{q},?),(\mathtt{r},2),(\mathtt{y},4)\}$ and that $\mathsf{RD}_{entry}(5)([])$ has value $\{(\mathtt{n},?),(\mathtt{q},?),(\mathtt{y},4)\}$. Then we can calculate the value of $\mathsf{RD}_{exit}(6)([])$ as follows.

$$
\begin{array}{ll}
 & \mathsf{RD}_{exit}(6)([]) \\
= & \quad \langle \text{by definition, since } \mathit{return}(5,6) \rangle \\
 & \widehat{f^2_{5,6}}(\mathsf{RD}_{entry}(5), \mathsf{RD}_{entry}(6))([]) \\
= & \quad \langle \text{by definition of } \widehat{f^2_{5,6}} \rangle \\
 & ((\{(x,\ell) \mid (x,\ell) \in \mathsf{RD}_{entry}(5)([]), x \in \mathit{formals}(6), \ell \in \mathbf{Lab}^?_\star\} \\
 & \quad \cup (\mathsf{RD}_{entry}(6)([] : 5) \setminus \{(x,\ell') \mid x \in \mathit{formals}(6), \ell' \in \mathbf{Lab}^?_\star\})) \\
 & \quad \setminus \{(z,\ell') \mid z \in \mathit{res\text{-}actuals}(6), \ell' \in \mathbf{Lab}^?_\star\}) \\
 & \cup \{(z,6) \mid z \in \mathit{res\text{-}actuals}(6)\} \\
= & \quad \langle \text{by the program and definitions of auxiliary functions} \rangle \\
 & ((\{(x,\ell) \mid (x,\ell) \in \mathsf{RD}_{entry}(5)([]), x \in \{\mathtt{n},\mathtt{r}\}, \ell \in \mathbf{Lab}^?_\star\} \\
 & \quad \cup (\mathsf{RD}_{entry}(6)([] : 5) \setminus \{(x,\ell') \mid x \in \{\mathtt{n},\mathtt{r}\}, \ell' \in \mathbf{Lab}^?_\star\})) \\
 & \quad \setminus \{(z,\ell') \mid z \in \{\mathtt{q}\}, \ell' \in \mathbf{Lab}^?_\star\}) \\
 & \cup \{(z,6) \mid z \in \{\mathtt{q}\}\} \\
= & \quad \langle \text{by assumptions} \rangle \\
 & ((\{(x,\ell) \mid (x,\ell) \in \{(\mathtt{n},?),(\mathtt{q},?),(\mathtt{y},4)\}, x \in \{\mathtt{n},\mathtt{r}\}, \ell \in \mathbf{Lab}^?_\star\} \\
 & \quad \cup (\{(\mathtt{n},5),(\mathtt{q},?),(\mathtt{r},2),(\mathtt{y},4)\} \setminus \{(x,\ell') \mid x \in \{\mathtt{n},\mathtt{r}\}, \ell' \in \mathbf{Lab}^?_\star\})) \\
 & \quad \setminus \{(z,\ell') \mid z \in \{\mathtt{q}\}, \ell' \in \mathbf{Lab}^?_\star\}) \\
 & \cup \{(z,6) \mid z \in \{\mathtt{q}\}\} \\
= & \quad \langle \text{by set theory} \rangle \\
 & ((\{(\mathtt{n},?)\} \cup \{(\mathtt{q},?)(\mathtt{y},4)\}) \\
 & \quad \setminus \{(z,\ell') \mid z \in \{\mathtt{q}\}, \ell' \in \mathbf{Lab}^?_\star\}) \\
 & \cup \{(z,6) \mid z \in \{\mathtt{q}\}\} \\
= & \quad \langle \text{by set theory} \rangle \\
 & \{(\mathtt{n},?),(\mathtt{y},4)\} \cup \{(\mathtt{q},6)\} \\
= & \quad \langle \text{by set theory} \rangle \\
 & \{(\mathtt{n},?),(\mathtt{q},6),(\mathtt{y},4)\}
\end{array}
$$

References

[1] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis.* Springer-Verlag, second printing edition, 2005.