**Sets of States at Each Program Point**

Consider the program

i = 0;

while (i < 10) {

if (i > 1)

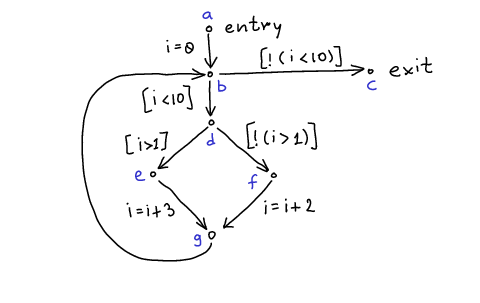
i = i + 3;

else

i = i + 2;

}

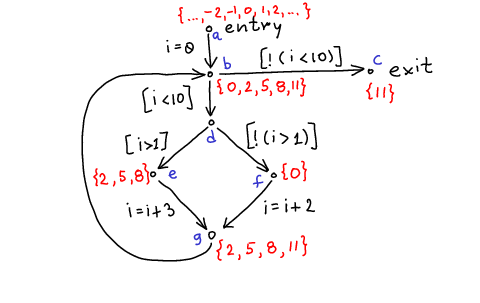
A possible corresponding control-flow graph is:

[](http://lara.epfl.ch/w/_detail/cc09:cfg-inc.png?id=cc09%3Asets_of_states_at_each_program_point&cache=cache)

Suppose that

* program state is given by the value of the integer variable i
* initially, it is possible that i has any value

Compute the set of states at each vertex in CFG

[](http://lara.epfl.ch/w/_detail/cc09:cfg-inc-states.png?id=cc09%3Asets_of_states_at_each_program_point&cache=cache)

For each program point $\{a,b,c,d,e,f,g\}$we associate a set of states $S(a)$, $S(b)$, ..., $S(g)$.

**Running the Program**

One way to describe the set of state: for each initial state, run the CFG with this state and insert the modified states at appropriate points.

**Reachable States as A Set of Recursive Equations**

If $c$is the label on the edge of the graph, let $R(c)$denote the relation between initial and final state that describes the meaning of statement. For example,   
\begin{displaymath}
\begin{array}{l}
  R(i=0) = \{ (i,i') \mid i' = 0 \}  \\
  R(i=i+2) = \{ (i,i') \mid i' = i+2 \}  \\
  R(i=i+3) = \{ (i,i') \mid i' = i+3 \}  \\
  R([i<10]) = \{ (i,i') \mid i'=i \land i < 10 \} \\
\end{array}
\end{displaymath}  
We will write $T(S,c)$for the image of set $S$under relation $R(c)$. For example,   
\begin{displaymath}
   T(\{ 10, 15, 20 \},i=i+2) = \{ 12, 17, 22 \}
\end{displaymath}  
Note that that if [p] is a condition (as generated by ‘if’ and ‘while’) then   
\begin{displaymath}
   T(S, [p]) = \{ x \in S \mid p \}
\end{displaymath}

If an edge has no label, we denote it skip. So, $T(S,skip)=S$.

Now, we can describe the meaning of our program using recursive equations:   
\begin{displaymath}
\begin{array}{l}
S(a) = \{ \ldots, -2, -1, 0, 1, 2, \ldots \} \\
S(b) = T(S(a),i=0) \cup T(S(g),skip) \\
S(c) = T(S(b),[\lnot(i<10)]) \\
S(d) = T(S(b),[i<10]) \\
S(e) = T(S(d),[i>1]) \\
S(f) = T(S(d),[\lnot(i>1)]) \\
S(g) = T(S(e),i=i+3) \cup T(S(f),i=i+2) 
\end{array}
\end{displaymath}

Note that the solution we computed satisfies the recursive equations. In fact, our solution is the unique **least** solution of these equations.

The problem with these exact equations is that they are as difficult to compute as running the program on all possible input states. Instead, when we do data-flow analysis, as with type systems, we will consider approximate descriptions of these sets of states.