A Proofs

Proof of Lemma 7.1. Observe that $S^k$ is monotonic in $k$. Hence the lemma is equivalent to the following stronger claim: if $A(u)(i)$ is defined, then there exists a $k$ such that $S^n(u)(i)$ is defined and equal to $A(u)(i)$, for all $n \geq k$. The proof is by induction on the program execution steps, i.e., $step(u, i)$, and is divided into a number of cases corresponding to the different types of vertices. In each case, the argument follows the following general outline:

1. If $A(u)(i)$ is defined, then program point $u$ executes at least $i$ times. From the properties observed earlier, $A(u)(i)$ is shown to be some function $f_u$ of the values computed at some other program points at particular instances:

   $$A(u)(i) = f_u(A_{u_1}(1\ldots i_1), A_{u_2}(1\ldots i_2), \ldots),$$

   where $step(u_j, i_j) < step(u, i)$, for all $j$.

2. From the inductive hypothesis, we assume the existence of a $k$ such that $S^k(u)(1\ldots i_j)$ is defined and equal to $A(u_j)(1\ldots i_j)$, for all $j$.

3. We then look at the definition of $S^{k+1}(u)$, obtained from the set of recursive equations,

   $$S^{k+1}(u) = F_u(S^k(v_1), S^k(v_2), \ldots),$$

   and show that $S^{k+1}(u)(i)$ is defined and equal to $f_u(A_{u_1}(1\ldots i_1), A_{u_2}(1\ldots i_2), \ldots)$, completing the proof.

Case 1: Let $u$ be the Start vertex or some Initialize vertex. This is the base case, and the proof is trivial. Under an appropriate interpretation of these vertices, $u$ executes only once. From the definition, we can easily verify that $S^1(u)(1)$ is defined and equal to $A(u)(1)$.

Case 2: Let $u$ be a FinalUse vertex. Let $v$ be its sole reaching definition. Both $u$ and $v$ can execute at most one time, and $v$ must execute before $u$. The result follows trivially.

Case 3: Let $u$ be a $\phi_T$ or $\phi_F$ vertex. Assume, without loss of generality, that $u$ is a $\phi_T$ vertex. Let $v$ denote $parent(u)$ and $w$ denote $dataPred(u)$. From property 11 in §6, $j = index(A(v), i, true)$ must be defined and

$$step(w, j) < step(v, j) < step(u, i) < step(w, j + 1)$$

and $A(u)(i)$ must be equal to $A(w)(j)$. From the inductive hypothesis, there exists a $k$ such that $S^k(w)(1\ldots j)$ is defined and equal to $A(w)(1\ldots j)$ and $S^k(v)(1\ldots j)$ is defined and equal to $A(v)(1\ldots j)$ (and, in particular, $index(S^k(v), i, true) = j$). By definition,

$$S^{k+1}(u) = select(true, S^k(v), S^k(w))$$

It is a property of $select$ that $S^{k+1}(u)(i)$ is defined and equal to $A(u)(i)$.

Case 4: Let $u$ be a $\phi_y$ vertex. Let $v$ be $ifNode(u)$, $x$ be $trueDef(u)$ and $y$ be $falseDef(u)$. Obviously, the $parent$ of both $x$ and $y$ is $v$. As observed in §6, $step(v, i) < step(u, i)$ (property 8), $step(x, j) < step(u, i)$ (property 12),

1
and \( \text{step}(y,i-j) < \text{step}(u,i) \) (property 12), where \( j = \#(A(v),i,true) \) and \( i-j = \#(A(v),i,false) \). Furthermore, from property 13,

\[
A(u)(i) = \begin{cases} 
  A(x)(j) & \text{if } A(v)(i) \\
  A(y)(i-j) & \text{otherwise}
\end{cases}
\]

From the inductive hypothesis, there exists a \( k \) such that \( S^k(v)(1\ldots i) = A(v)(1\ldots i) \), \( S^k(x)(1\ldots j) = A(x)(1\ldots j) \), and \( S^k(y)(1\ldots i-j) = A(y)(1\ldots i-j) \), while from the definition,

\[
S^{k+1}(u) = \text{merge}(S^k(v),S^k(x),S^k(y))
\]

It follows that \( S^{k+1}(u)(i) \) is defined and equal to \( A(u)(i) \), as required.

Case 5: Let \( u \) be a \( \phi_{\text{Ext}} + \phi_{\text{while}} \) vertex. As can be seen from the defining equations in these cases, these are similar to \( \phi_T \) and \( \phi_T \) vertices, and the proof is similar, too.

Case 6: Let \( u \) be a \( \phi_{\text{Ext}} \) vertex. Let \( v, x, \) and \( y \) be whileNode\((u)\), outer\(\text{Def}(u)\), and inner\(\text{Def}(u)\), respectively. Let \( w \) be the parent of \( x \) and \( v \). Assume, without loss of generality, that the control dependences \( w \to v \) and \( w \to u \) are labeled true. Consider the case \( i = 1 \) first. We showed in §6 (property 14) that \( \text{step}(x,1) < \text{step}(u,1) \), and that \( A(u)(1) \), if defined, must be equal to \( A(x)(1) \). Consider \( i > 1 \). Again, we showed that \( \text{step}(v,i-1) < \text{step}(u,i) \), \( \text{step}(x,j) < \text{step}(u,i) \), and \( \text{step}(y,i-j) < \text{step}(u,i) \), where \( j = \#(A(v),i-1,false) + 1 \).

Furthermore,

\[
A(u)(i) = \begin{cases} 
  A(y)(i-j) & \text{if } A(v)(i-1) \\
  A(x)(j) & \text{otherwise}
\end{cases}
\]

The hypothesis implies the existence of a \( k \) such that \( S^k(v)(1\ldots i-1) = A(v)(1\ldots i-1) \), \( S^k(y)(1\ldots i-j) = A(y)(1\ldots i-j) \), and \( S^k(x)(1\ldots j) = A(x)(1\ldots j) \). By definition,

\[
S^{k+1}(u) = \text{whileMerge}(S^k(v),S^k(y),S^k(x))
\]

The properties of \( \text{whileMerge} \) imply that \( S^{k+1}(u)(i) \) is defined and equal to \( A(u)(i) \).

Case 7: Let \( u \) be a \( \phi_{\text{copy}} \) vertex. The proof is similar to the above one, simplified by the fact that there is no definition of \( \text{varDef}(u) \) inside the loop. Let \( v \) denote whileNode\((u)\), and \( w \) denote data\(\text{Pred}(u)\). We showed in §6 (property 15) that \( \text{step}(v,i-1) < \text{step}(u,i) \), \( \text{step}(w,j) < \text{step}(u,i) \), where \( j = \#(A(v),i-1,false) + 1 \), and that \( A(u)(i) \) must be equal to \( A(w)(j) \). From the hypothesis, there exists a \( k \) such that

\[
S^k(v)(1\ldots i-1) = A(v)(1\ldots i-1)
\]

and

\[
S^k(w)(1\ldots j) = A(w)(1\ldots j)
\]

and by definition

\[
S^{k+1}(u) = \text{whileCopy}(S^k(v),S^k(w))
\]
It follows that $S^{k+1}(u)(i)$ is defined and equal to $A(u)(i)$, as required.

Case 8: Let $u$ be an assignment statement, if predicate, or while predicate, and let $u$ have at least one data-dependence predecessor. Let $u_1, u_2, \ldots, u_n$ represent the $n$ data-dependence predecessors of $u$. We know that $\text{step}(u_j, i) < \text{step}(u, i)$ for all $j \leq n$ (property 8), and that $A(u)(i)$ must be equal to $\text{functionOf}(u)(A(u_1)(i_1), \ldots, A(u_n)(i_n))$ (property 9). From the inductive hypothesis, there exists a $k$ such that, for $1 \leq j \leq n$,

$$S^k(u_j)(1 \ldots i) = A(u_j)(1 \ldots i)$$

By definition,

$$S^{k+1}(u) = \text{map}(\text{functionOf}(u))(S^k(u_1), \ldots, S^k(u_n))$$

It follows that $S^{k+1}(u)(i)$ is defined and equal to $A(u)(i)$.

Case 9: Let $u$ be a constant-valued assignment statement or if predicate. Let $v$ be $u$’s parent. Assume, without loss of generality, that the control dependence $v \rightarrow_c u$ is labeled true. We know from property 10 of §6 that $j = \text{index}(A(v), i, \text{true})$ must be defined and that

$$\text{step}(v, j) < \text{step}(u, i)$$

Hence, there exists a $k$ such that $S^k(v)(1 \ldots j)$ is defined and equal to $A(v)(1 \ldots j)$. By definition,

$$S^{k+1}(u) = \text{replace}(\text{true}, c, S^k(v))$$

and the required result follows.

Case 10: Let $u$ be a constant-valued while predicate. If the constant is false, the vertex behaves just like vertices in the previous case. If the constant is true, and if $u$ executes at least once, then there must be a $k$ and $j$ such that $S^k(v)(j)$ is defined and the same as $\text{label}(v, u)$, where $v$ is $u$’s parent. From the definition, it can be seen that $S^{k+1}(u)$ is an infinite sequence of trues, satisfying the requirement.

We have proved the lemma for each possible value of $\text{typeOf}(u)$, and hence the lemma follows.

**Proof of Lemma 7.3.** The proof is by induction on $k$. Assume that the program terminates normally and that $S^k(u)(i)$ is defined. We show that $A(u)(i)$ is defined. The equality of $A(u)(i)$ and $S^k(u)(i)$ then follows from the previous lemma and the fact that $S^k(u)$ is monotonic in $k$.

Now, $A(u)(i)$ is defined iff $u$ executes $i$ times. Thus, it is enough to show that $u$ executes $i$ times, which we do below. (Similarly, the inductive hypothesis may be interpreted as: if $S^{k-1}(v)(j)$ is defined, then $A(v)(j)$ is defined and, hence, $v$ must have executed $j$ times.)

Case 1: Let $u$ be the Start vertex or some Initialize vertex. The proof is trivial in this case.

Case 2: Let $u$ be a FinalUse vertex. Let $v$ denote dataPred$(u)$. By definition, $S^k(u) = S^{k-1}(v)$. Thus, if $S^k(u)(i)$ is defined, then so is $S^{k-1}(v)(i)$. From the inductive hypothesis, program point $v$ must have executed $i$ times (which also
means that \( i \) must be 1, but that is immaterial). Since \( u \) and \( v \) have the same control-dependence predecessors, \( u \) must also execute \( i \) times (before the program can terminate normally).

Case 3: Let \( u \) be a \( \phi_T \) vertex. Let \( v \) denote \( ifNode(u) \) and \( w \) denote \( dataPred(u) \). By definition, \( S^k(u) = select(true, S^{k-1}(v), S^{k-1}(w)) \) Hence, if \( S^k(u)(i) \) is defined, then \( S^{k-1}(v) \) must contain at least \( i \) true values. The hypothesis implies that \( v \) must have evaluated to true at least \( i \) times. Hence \( u \) must execute for an \( i^{th} \) time. The proof is similar for a \( \phi_F \) vertex.

Case 4: Let \( u \) be a \( \phi_y \) vertex. Let \( w,x, \) and \( y \) denote \( ifNode(u), trueDef(u) \), and \( falseDef(u) \), respectively. Then, \( S^k(u) = select(false, S^{k-1}(v), S^{k-1}(w)) \). If \( S^k(u)(i) \) is defined, then \( S^{k-1}(v)(i) \) must also be defined. The hypothesis implies that \( w \) must have executed \( i \) times. Consequently, \( u \) must also have executed \( i \) times.

Case 5: Let \( u \) be a \( \phi_{Exit} \) vertex. Let \( v \) and \( w \) denote \( whileNode(u) \) and \( dataPred(u) \), respectively. Then, \( S^k(u) = select(false, S^{k-1}(v), S^{k-1}(w)) \). If \( S^k(u)(i) \) is defined, then \( S^{k-1}(v) \) must contain at least \( i \) occurrences of false. From the inductive hypothesis, the corresponding while loop must have completed execution at least \( i \) times. Hence \( u \) must have executed at least \( i \) times.

Case 6: Let \( u \) be a \( \phi_{Enter} \) vertex. The proof is similar to the case of a \( \phi_T \) vertex.

Case 7: Let \( u \) be a \( \phi_{Enter} \) vertex. Let \( v, y \), and \( x \) denote \( whileNode(u), innerDef(u) \), and \( outerDef(u) \), respectively. Then, \( S^k(u) = whileMerge(S^{k-1}(v), S^{k-1}(y), S^{k-1}(x)) \). Consider the case \( i = 1 \). If \( S^k(u)(1) \) is defined, then \( S^{k-1}(x)(1) \) must be defined, too. Hence, \( x \) must have executed at least once, from the induction hypothesis. Consequently, \( u \) must have executed at least once, too. Consider the case \( i > 1 \). If \( S^k(u)(i) \) is defined, \( S^{k-1}(v)(1 \ldots i - 1) \) must be defined, too. Consequently, \( v \) must have executed \( i - 1 \) times, by the induction hypothesis. Suppose it evaluated to true in the \( i - 1^{th} \) time, i.e., assume \( S^{k-1}(v)(i - 1) \) were true. Then \( u \) must subsequently execute, for an \( i^{th} \) time. On the other hand, let \( S^{k-1}(v)(i - 1) \) be false. Let \( j = \#(S^{k-1}(v), i - 1, false) \). Then, \( S^{k-1}(x)(j + 1) \) must be defined. That is, \( x \) must have executed at least once after \( u \) had executed \( i - 1 \) times. Hence, \( u \) must execute for an \( i^{th} \) time, too.

Case 8: Let \( u \) be a \( \phi_{Copy} \) vertex. The proof is just as in the previous case.

Case 9: Let \( u \) be an \( assignment \), if predicate, or \( while \) predicate, with \( n \) data-dependence predecessors \( u_1 \ldots u_n \), where \( n > 0 \). Then, \( S^k(u) = map(f)(S^{k-1}(u_1), \ldots, S^{k-1}(u_n)) \). If \( S^k(u)(i) \) is defined, then \( S^{k-1}(u_j)(i) \) must be defined, for all \( j \). Thus, \( u_j \) must have executed \( i \) times. Hence, \( u \) must also execute \( i \) times, because \( u \) and all the \( u_j \) have the same control-dependence predecessors.

Case 10: Let \( u \) be a constant-valued assignment statement or if predicate. Let \( v \) be \( u \)'s parent. Assume, without loss of generality, that the control dependence \( v \rightarrow_c u \) is labeled true. Then \( S^k(u) = replace(true, functionOf(u), S^{k-1}(v)) \). Thus, if \( S^k(u)(i) \) is defined, then \( S^{k-1}(v) \) must contain at least \( i \) occurrences of true. Hence, \( v \) must have evaluated to true at least \( i \) times. So, \( u \) must execute at least \( i \) times.

4
Case 11: Let $u$ be a constant-valued *while* predicate. If the constant is *false*, the vertex behaves like the vertices in the previous case. Otherwise, if $S^k(u)(i)$ is defined, then its parent $v$ must have evaluated to $\text{label}(v, u)$ at least once, which would have caused $u$ to execute. This would have resulted in an infinite loop, contradicting the assumption that the program halts. Hence $S^k(u)$ must be a null sequence, for any $k$, completing the proof.