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., $\mathit{step}(u,i)$, 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:

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

   where $\mathit{step}(u_j, i_j) < \mathit{step}(u,i)$, for all j.
2. From the inductive hypothesis, we assume the existence of a k such that $S^k(u_j)(1 \dots i_j)$ is defined and equal to $A(u_j)(1 \dots i_j)$, for all j.
3. We then look at the definition of $S^{k+1}(u)$, obtained from the set of recursive equations,

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

   and show that $S^{k+1}(u)(i)$ is defined and equal to $f_u(A_{u_1}(1 \dots i_1), A_{u_2}(1 \dots i_2), \dots)$, 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 ϕ_T or ϕ_F vertex. Assume, without loss of generality, that u is a ϕ_T vertex. Let v denote $\mathit{parent}(u)$ and w denote $\mathit{dataPred}(u)$. From property 11 in §6, $j = \mathit{index}(A(v), i, \mathit{true})$ must be defined and

$$\mathit{step}(w,j) < \mathit{step}(v,j) < \mathit{step}(u,i) < \mathit{step}(w,j+1)$$

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

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

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 ϕ_{if} vertex. Let v be $\mathit{ifNode}(u)$, x be $\mathit{trueDef}(u)$ and y be $\mathit{falseDef}(u)$. Obviously, the *parent* of both x and y is v. As observed in §6, $\mathit{step}(v,i) < \mathit{step}(u,i)$ (property 8), $\mathit{step}(x,j) < \mathit{step}(u,i)$ (property 12), and $\mathit{step}(y, i-j) < \mathit{step}(u,i)$ (property 12), where $j = \#(A(v), i, \mathit{true})$ and $i - j = \#(A(v), i, \mathit{false})$. 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,

$$S^{k+1}(u) = \mathit{merge}(S^k(v), S^k(x), S^k(y))$$

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

Case 5: Let u be a ϕ_{Exit} or ϕ_{while} vertex. As can be seen from the defining equations in these cases, these are similar to ϕ_F and ϕ_T vertices, and the proof is similar, too.

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

$$S^{k+1}(u) = \mathit{whileMerge}(S^k(v), S^k(y), S^k(x)).$$

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

Case 7: Let u be a ϕ_{copy} vertex. The proof is similar to the above one, simplified by the fact that there is no definition of $\mathit{varOf}(u)$ inside the loop. Let v denote $\mathit{whileNode}(u)$, and w denote $\mathit{dataPred}(u)$. We showed in §6 (property 15) that $\mathit{step}(v, i-1) < \mathit{step}(u, i)$, $\mathit{step}(w, j) < \mathit{step}(u, i)$, where $j = \#(A(v), i-1, \mathit{false}) + 1$, 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

$$S^{k+1}(u) = \mathit{whileCopy}(S^k(v), S^k(w))$$

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 $\mathit{step}(u_j, i) < \mathit{step}(u, i)$ for all $j \leq n$ (property 8), and that $A(u)(i)$ must be equal to $\mathit{functionOf}(u)(A(u_1)(i_1), \ldots, A(u_n)(i_n))$ (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,

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

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 $j = \mathit{index}(A(v), i, \mathit{true})$ must be defined and that

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

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,

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

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 $\mathit{label}(v, u)$, where v is u's parent. From the definition, it can be seen that $S^{k+1}(u)$ is an infinite sequence of *true*s, satisfying the requirement.

We have proved the lemma for each possible value of $\mathit{typeOf}(u)$, 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 $\mathit{dataPred}(u)$. 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 ϕ_T vertex. Let v denote $\mathit{ifNode}(u)$ and w denote $\mathit{dataPred}(u)$. By definition, $S^k(u) = \mathit{select}(\mathit{true}, S^{k-1}(v), S^{k-1}(w))$ 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 ϕ_F vertex.

Case 4: Let u be a ϕ_{if} vertex. Let w,x, and y denote $\mathit{ifNode}(u)$, $\mathit{trueDef}(u)$, and $\mathit{falseDef}(u)$, respectively. Then, $S^k(u) = \mathit{merge}(S^{k-1}(w), S^{k-1}(x), S^{k-1}(y))$. If $S^k(u)(i)$ is defined, then $S^{k-1}(w)(i)$ 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 ϕ_{Exit} vertex. Let v and w denote $\mathit{whileNode}(u)$ and $\mathit{dataPred}(u)$, respectively. Then, $S^k(u) = \mathit{select}(\mathit{false}, S^{k-1}(v), S^{k-1}(w))$. 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 ϕ_{while} vertex. The proof is similar to the case of a ϕ_T vertex.

Case 7: Let u be a ϕ_{Enter} vertex. Let v, y, and x denote $\mathit{whileNode}(u)$, $\mathit{innerDef}(u)$, and $\mathit{outerDef}(u)$, respectively. Then, $S^k(u) = \mathit{whileMerge}(S^{k-1}(v), S^{k-1}(y), S^{k-1}(x))$. 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 $j = \#(S^{k-1}(v), i-1, \mathit{false})$. 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 ϕ_{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, $S^k(u) = \mathit{map}(f)(S^{k-1}(u_1), \ldots, S^{k-1}(u_n))$. 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 $S^k(u) = \mathit{replace}(\mathit{true}, \mathit{functionOf}(u), S^{k-1}(v))$. 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.

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 $\mathit{label}(v, u)$ 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.