

```

1: Variables
2:   code is WyIL function call code at function f.
3:   caller is caller site of a function call.
4:   callee is callee site of a function call.
5: end Variables
6: // Propagate parameter bounds to callee
7: procedure PROPAGATE_INPUTBOUND(callee, param_bounds)
8:   entry := callee.entry // Get entry block in callee
9:   for each param in params do
10:     // Propagate parameter bound as Range constraint
11:     r := Map parameter to register r at callee
12:     param_bound := get bounds of parameter param from param_bounds
13:     entry.add(Range(param_bound, r))
14:   end for
15: end procedure
16: // Propagate return bounds back to caller
17: procedure PROPAGATE_OUTPUTBOUND(caller, code, ret_bounds)
18:   blk := caller.getblk(code) // Get the block of function call code at caller
19:   r := function return r at caller
20:   // Propagate return bounds as Range constraint at caller
21:   blk.add(Range(ret_bounds, r))
22: end procedure
23: // Infer function call code at function f
24: procedure INFER_FUNCTIONCALL(code, f)
25:   caller := f // Caller is function f
26:   callee := code.callee // Get callee from function call code
27:   // Infer parameter bounds at caller
28:   param_bounds := INFER_FUNCTION(caller)
29:   // Propagate parameter bounds to callee
30:   PROPAGATE_INPUTBOUND(callee, param_bounds)
31:   // Infer return bounds at callee
32:   ret_bounds := INFER_FUNCTION(callee)
33:   // Propagate return bounds back to caller
34:   PROPAGATE_OUTPUTBOUND(caller, code, ret_bounds)
35: end procedure

```